IMPLEMENTASI ALGORITME SPECK UNTUK ENKRIPSI DAN DEKRIPSI PADA QR *CODE*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Yuniar Siska Fatmala
NIM:145150207111054

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI ALGORITME SPECK UNTUK ENKRIPSI DAN DEKRIPSI PADA QR *CODE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Yuniar Siska Fatmala
NIM: 145150207111054

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Ari Kusyanti, S.T, M.Sc	Mahendra Data, S.Kom., M.Kom
NIK.201102 831228 2 001	NIK:201503 861117 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 27 Juli 2018

Yuniar Siska Fatmala

NIM: 145150207111121

KATA PENGANTAR

Alhamdulillah, Puji syukur kehadirat Allah SWT Yang Maha Mendengar lagi Maha Melihat dan atas segala limpahan rahmat, taufik, serta hidayah-Nya sehingga penulis dapat menyelesaikan karya tulis yang berbentuk skripsi ini sesuai dengan waktu yang telah direncanakan. Shalawat serta salam semoga senantiasa tercurahkan kepada baginda Nabi Besar Muhammad SAW beserta seluruh keluarga dan sahabatnya yang selalu eksis membantu perjuangan beliau dalam menegakkan Dinullah di muka bumi ini. Sehingga tugas akhir yang berjudul "IMPLEMENTASI ALGORITME SPECK UNTUK ENKRIPSI DAN DEKRIPSI PADA QR-*CODE*" dapat diselesaikan. Penulisan skripsi ini dilakukan sebagai kewajiban penulis dalam menempuh tugas akhir untuk menyelesaikan studi Strata S1 dan memperoleh gelar Sarjana Komputer (S.Kom) pada program Sarjana Fakultas Ilmu Komputer Universitas Brawijaya.

Penulis menyadari bahwa dalam penyusunan Skripsi ini belum sempurna dan mungkin masih banyak kekurangan – kekurangannya. Karena itu, saran dan kritik akan di terima dengan senang hati guna untuk membangun demi penulisan laporan yang lebih baik.Penulisan laporan skripsi ini dapat terselesaikan atas bantuan serta dorongan orang-orang yang berjasa kepada penulis. Untuk itu, penulis sangat berterima kasih kepada pihak-pihak yang telah membantu baik dalam hal solusi, doa, izin, serta dukungan moral sehingga penulis dapat menyelesaikan skripsi ini. Dengan ini penulis berterima kasih kepada:

1. Allah SWT yang telah memberikan rahmat, karunia, petunjuk, serta pertolongan-Nya kepada penulis.

2. Bapak wayan Firdaus mahmudy, S.Si, M.T, Ph.D selaku dekan fakultas ilmu komputer universitas brawijaya.

3. Ibu Ari Kusyanti, S.T, M.Sc selaku dosen pembimbing I yang telah membimbing dan memberikan dukungan kepada penulis dalam menyusun skripsi ini.

4. Bapak Mahendra Data, S.Kom., M.Kom selaku dosen pembimbing II yang telah membimbing dan memberikan dukungan kepada penulis dalam menyusun skripsi ini.

5. Kedua orang tua saya, Bapak H.Masroni, S.Ag, M.Pd.I dan Ibu Hj. Dewi Afidah Amd.Keb yang telah memberikan dukungan dan doa yang tiada habisnya kepada saya selaku putri keduanya sehingga saya dapat menyelesaikan skripsi ini dengan luar biasa.

6. Kakak saya drg.Nikmatul Amaliya Nur Cahyani dan Bayu Pandu Wibisono, S.Kom yang memberikan motivasi dan dukungan dalam menyelesaikan skripsi ini.

7. Sahabat - sahabat saya yaitu Keke Eka Pratama, Dwi Qunita Putri, Mahdarani Dwi Laxmi, Nuriya Fadilah, Maya Rowiyatun dll. Yang telah memberikan dukungan, semangat dan bantuan kepada saya selaku penulis.

Tentunya sebagai manusia tidak pernah luput dari kesalahan, penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, Oleh karena itu saran dan kritik yang konstruktif dari semua pihak sangat diharapkan demi penyempurnaan selanjutnya. Akhirnya hanya kepada Allah SWT kita kembalikan semua urusan dan semoga skripsi ini dapat bermanfaat bagi semua pihak, khususnya bagi penulis dan para pembaca pada umumnya. Semoga Allah SWT meridhoi dan dicatat sebagai ibadah disisi-Nya, amin.

Malang, 27 Juli 2018

Penulis

yuniarsiskafatmala@gmail.com

ABSTRAK

Yuniar Siska Fatmala, Implementasi Algoritme SPECK untuk Enkripsi dan Dekripsi pada QR *Code*.

Dosen Pembimbing: Ari Kusyanti dan Mahendra Data

Abstrak

Seiring dengan berkembangnya teknologi, muncul teknologi baru yang diprediksi dapat menggantikan fungsi dari *barcode*. Teknologi tersebut adalah teknologi QR *code*. *Quick Response Codes* atau QR *codes* merupakan bar*codes* dua dimensi yang dikembangkan oleh *Denso Wave Corporation* pada tahun 1994. QR *Code* mampu menyimpan informasi yang cukup besar oleh karena itu *Qr Code* perlu diamankan kareana saat ini banyak sekali pembajakan sebuah kode, yang nantinya akan merugikan seseorang atau sebuah instansi yang berkaitan. Ada beberapa upaya dalam menjaga kerahaasiaan data agar tidak disalah gunakan, Salah satunya adalah dengan pengimplementasian algoritme SPECK. SPECK adalah keluarga *blokcipher* ringan yang diterbitkan oleh Badan Keamanan Nasional A.S. pada tahun 2013. Dari hasil implementasi algoritme SPECK dapat disimpulkan bahwa aplikasi ini dapat mengenkripsi semua jenis karakter berupa *string*, huruf, angka ataupun simbol yang perlu untuk diamankan, dan pada saat mendekripsi QR *Code* aplikasi akan mengaktifkan fungsi kamera dan melakukan *scanning QR Code* yang akan menjadi *plainntext* kembali. Hal ini dibuktikan pada pengujian keamanan yang membuktikan bahwa hasil *ciphertext* yang sangat berbeda dengan *plaintext*. Waktu eksekusi enkripsi dan dekripsi dapat dibuktikan dengan pengujian kinerja waktu enkripsi dengan total kinerja waktu yang dibutuhkan yaitu 0.061614036560059 detik dan Pada pengujian kinerja waktu dekripsi total kinerja waktu yang dibutuhkan 0.0039241313934326 detik.

Kata kunci: *SPECK, Enkripsi, Dekripsi, Test Vector, blockcipher, QR Code.*

ABSTRACT

Yuniar Siska Fatmala, SPECK *Algorithm Implementation for Encryption and Decryption on QR Code.*

***Supervisor*: Ari Kusyanti and Mahendra Data**

Abstract

Along with the development of technology, emerging new technologies are predicted to replace the function of the barcode. The technology is QR code technology. Quick Response Codes or QR codes are two-dimensional barcodes developed by Denso Wave Corporation in 1994. The QR Code is capable of storing information that is large enough therefore the QR Code needs to be secured because currently there is a lot of piracy of a code, which will harm someone or a related agency. There are several efforts to maintain data confidentiality so that it is not misused, one of which is by implementing the SPECK algorithm. SPECK is a family of lightweight blockcipers published by the U.S. National Security Agency. in 2013. From the results of the SPECK algorithm implementation can be concluded that this application can encrypt all types of characters in the form of strings, letters, numbers or symbols that need to be secured, and when decrypting the QR Code the application will activate the camera function and scan the QR Code that will become plainntext again. This is evidenced in security testing which proves that the results of ciphertext are very different from plaintext. The execution time of encryption and decryption can be proven by testing the performance of encryption time with the total performance required time is 0.061614036560059 seconds and in testing the performance of the decryption time total performance of the time taken is 0.0039241313934326 seconds.

Keywords: SPECK, *Encryption, Decryption, Test Vector, blockcipher, QR Code.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Indonesia yang saat ini telah masuk pada era teknologi, dimana hampir segala sesuatunya dilakukan dengan komputasi dan perangkat. Perkembangan teknologi menyentuh hampir setiap sisi kehidupan masyarakat di era ini. Kemajuan teknologi adalah sesuatu yang tidak bisa dihindari dalam kehidupan ini, karena kemajuan teknologi akan berjalan sesuai dengan kemajuan ilmu pengetahuan. Setiap inovasi diciptakan untuk memberikan manfaat positif bagi kehidupan manusia. Teknologi juga memberikan banyak kemudahan, serta sebagai cara baru dalam melakukan aktivitas manusia. Manusia juga sudah menikmati banyak manfaat yang dibawa oleh inovasi-inovasi teknologi yang telah dihasilkan dalam dekade terakhir ini. Pada era globalisasi saat ini, penguasaan teknologi menjadi prestise dan indikator kemajuan suatu negara (Ngafifi, 2014).

Seiring dengan berkembangnya teknologi, muncul teknologi baru yang diprediksi dapat menggantikan fungsi dari *barcode*. Teknologi tersebut adalah teknologi QR *code*. *Quick Response Codes* atau QR *codes* merupakan bar *codes* dua dimensi yang dikembangkan oleh *Denso Wave Corporation* pada tahun 1994. Karakteristik utama dari QR *code* dibandingkan dengan *barcode* tradisional adalah baik posisi vertikal maupun horisontal dapat digunakan untuk menyimpan data sedangkan *barcode* biasa hanya dapat menyimpan informasi pada satu posisi saja. Karena alasan inilah QR *code* dapat menyimpan informasi lebih besar. *Barcode* biasanya hanya mampu maksimal menyimpan 20 digit informasi, sedangkan QR *code* dapat menyimpan 4296 karakter *alfanumeris*.Untuk bisa mengakses informasi dalam QR *code* membutuhkan kamera ponsel dan aplikasi perangkat lunak (Indriasari & Rahayu, 2012). Kamera ponsel digunakan untuk membaca QR *code* dimana posisi pembacaan tidak akan mempengaruhi gagal tidaknya proses baca QR *code*. Kamera ponsel membutuhkan perangkat lunak pembacaan QR code untuk mendecode QR *code*. Sistem operasi seperti Android dan Nokia Symbian biasanya sudah memiliki aplikasi QR *reader*. Untuk ponsel dengan sistem operasi yang lain bisa mendapatkan QR reader yang dibutuhkan di Internet. Penggunaan QR *code* saat ini sudah meluas digunakan sebagai alat promosi seperti kartu nama elektrik, brosur, kartu pos, *billboards*, bahkan pada layar televisi; yang dapat diakses oleh siapa saja dan dimana saja. Kelebihan QR *code* yang dapat menyimpan informasi yang cukup besar dimanfaatkan untuk mengkodekan data seperti teks, angka, gambar, *hyperlink*, dan lain-lain sesuai keperluan. Melihat kelebihan dari teknologi QR *code*, pemanfaatan QR *code* berpotensi untuk dapat digunakan dalam banyak bidang kehidupan. Salah satunya adalah dalam bidang layanan pada perpustakaan. (Indriasari & Rahayu, 2012).

Saat ini teknologi telah berkembang pesat. Salah satunya adalah teknologi *labelling*. Teknologi labelling yang sedang berkembang saat ini adalah QR Code. QR Code merupakan bentuk evolusi dari kode batang dari satu dimensi menjadi dua dimensi yang dikembangkan oleh *Denso Wave*. Pengenalan pola dilakukan dengan mendeteksi marker atau tanda yang telah diisi dengan informasi yang dibutuhkan. QR merupakan singkatan dari *Quick Response*. Tujuannya adalah untuk menyampaikan informasi dengan cepat dan mendapatkan respon yang cepat pula. Berbeda dengan kode batang yang hanya menyimpan informasi secara *horizontal*, QR-*code* mampu menyimpan informasi secara *horizontal* dan vertikal. Oleh karena itu, QR *Code* dapat menampung informasi yang lebih banyak misalnya dalam bentuk URL, *teks*, angka, dll. Karenanya, dengan menggunakan QR *Code*, kita dapat menyimpan informasi mengenai nomor surat, klasifikasi dokumen, pengirim dan perihal (QR Code.Com, 2010). Kelebihan QR *Code* dibandingkan dengan Barcode adalah: (1) kapasitas atau panjang kata lebih banyak; (2) tipe data yang disimpan pada QR *Code* beragam dapat berupa angka atau huruf atau gabungan keduanya; (3) QR *Code* dapat dibaca dari segala arah sehingga kemungkinan gagal dalam membaca sangat kecil; (4) memiliki ketahanan hingga 30%. Sehingga apabila QR *Code* mengalami kerusakan hingga 30 % dapat tetap terbaca (QR Code.Com, 2010).

Hal tersebut diantaranya juga mempengaruhi perkembangan teknologi dimana saat ini sudah banyak sekali pembajakan sebuah kode, yang nantinya akan merugikan seseorang atau sebuah instansi yang berkaitan, kode tersebut bisa berupa kode ujian, ataupun kode-kode yang lainnya yang perlu untuk diamankan, untuk itu itu diperlukan sebuah metode khusus dalam pengamanannya agar dapat meningkatkan kerahasiaan informasi (Hastaka & Mulyanto, 2012). Kerahasiaan merujuk pada perlindungan informasi dari penyingkapan pihak yang tidak sah. Dapat diperoleh dengan memberi akses terbatas pada informasi atau dengan penyandian informasi sehingga tidak memiliki arti apapun bagi pihak yang tidak berhak tersebut. Jika kerahasiaan ini tidak terpenuhi mengakibatkan adanya penyalahgunaan wewenang oleh pihak yang tidak sah (Syamsu, 2013). Ada beberapa upaya dalam menjaga kerahaasiaan data agar tidak disalah gunakan. Salah satunya adalah dengan pengimplementasian algoritme SPECK. SPECK adalah keluarga *blokcipher* ringan yang diterbitkan oleh Badan Keamanan Nasional A.S. pada tahun 2013. Keluarga SPECK terdiri dari 10 versi, mendukung berbagai ukuran blok dan kunci(Mohanty & M, *A Novel SPECK Algorithm for Faster Image Compression. International Conference on Machine Intelligence Research and Advancement*, 2013). *A Novel SPECK Algorithm forFaster Image Compression Filter* yang diusulkan digunakan melalui dekomposisi dengan penambahan tahap pra-proses untuk algoritme SPECK untuk kompresi citra uji standar. Modifikasi tersebut memberikan PSNR secara signifikan lebih tinggi. Skema pra-pemrosesan yang diusulkan, yang disebut *Zero-Shifting*, membawa nilai DC dalam kisaran integer yang ditandatangani serupa dengan nilai AC, sehingga perhitungan koefisien yang

berubah menjadi lebih konsisten. (Mohanty & Mohanty, *A Novel SPECK Algorithm for Faster Image Compression. International Conference on Machine Intelligence Research and Advancement,* 2013) menyimpulkan peningkatan yang signifikan dalam pengkodean dan penguraian waktu dengan membuat *codec* lebih cepat daripada yang asli. Namun, penelitian tersebut belum diterapkan pada studi kasus QR *Code* untuk mengenkripsi dan mendekripsi semua kalimat dalam bentuk String yang berupa karakter berbentuk kalimat . Pada penelitian sebelumnya, (MItra & Rakesh, 2016) meneliti *Desktop Application of QR Code for Data Security and Authentication.* Pada aplikasi ini menggunakan AES sebagai enkripsi. Sebagai kode QR dapat disesuaikan atau diformat sesuai keinginan perusahaan dengan menerapkan warna, embedding logo image atau label, maka akan membantu organisasi untuk mengkomunikasikan informasi dengan cara yang lebih menekankan dan membedakan daripada yang lain. Kode QR yang aman akan membantu perusahaan untuk mengkomunikasikan informasi dengan aman sehingga menghasilkan tujuan ganda pertukaran data yang aman dan tidak jelas dan akan membantu perusahaan menerapkan solusi untuk pertukaran informasi yang terkompresi, hemat biaya, aman dan tersembunyi. Namun, penelitian tersebut belum menerapkan alagortima SPECK.

Dalam penerapan algoritme SPECK diatas, proses enkripsi dan dekripsi dilakukan. Fungsinya adalah agar data yang disimpan dalam *database* diubah sedemikian rupa sehingga tidak mudah dibaca. Jadi enkripsi adalah proses yang dilakukan untuk merubah suatu informasi sehingga tidak dapat dibaca oleh orang yang tidak bertanggung jawab. Sebaliknya, proses dekripsi merupakan suatu proses yang mengembalikan informasi yang sudah dienkripsi menjadi bisa dibaca (Soewito, 2013).Metode keamanan tersebut dapat diterapkan pada kode QR. Qr *Code* digunakan dalam penelitian ini karena android merupakan OS (Operating System) mobile yang sangat populer dan banyak digunakan dan QR *Code* merupakan media yang digunakan dalam penyampaian informasi secara cepat dan mendapat respons yang cepat tanpa melakukan input secara manual dengan cara mengetik. Informasi yang dikodekan dalam QR *Code* dapat berupa URL, nomor telepon, pesan SMS, V-Card, atau teks apapun Penggunaan sistem pengamanan dengan menggunakan sandi klasik yang memiliki konsep relative sederhana dan banyak digunakan sampai sekarang yaitu vigener cipher akan membuat sistem ini memiliki sistem pengamanan yang tidak dengan mudah dipalsukan. Modifikasi yang sedemikian rupa yang digabungkan dengan QR-*Code* menjadikan sandi ini menjadi simple dan rumit dalam pemecahannya (Ashford, 2010). Berdasarkan alasan yang telah dipaparkan di atas, pada penelitian ini dibuat implementasi algoritme speck untuk enkripsi dan dekripsi pada QR *Code.*

1.2 Rumusan masalah

Dengan latar belakang di atas, maka rumusan masalah dari penulisan skripsi ini dapat diuraikan sebagai berikut:

1. Bagaimana cara SPECK mengimplementasi pada QR *Code*?
2. Bagaimana validasi plaintext dan *ciphertext* pada algoritme SPECK?
3. Bagaimana kinerja pemrosesan enkripsi dan dekripsi algoritme SPECK pada QR *Code*?

1.3 Tujuan

Tujuan dari penelitian ini dapat diuraikan sebagai berikut:

1. Mengetahui cara mengimplementasi SPECK pada QR *Code*
2. Mengetahui cara validasi *plaintext* dan *ciphertext* pada algoritme SPECK
3. Mengetahui kinerja pemrosesan enkripsi dan dekripsi algoritme SPECK pada QR *Code*

1.4 Manfaat

Manfaat yang diharapkan dari penelitian Implementasi Algoritme SPECK untuk Enkripsi dan Dekripsi pada QR *Code* ini diantaranya adalah:

1. Pengguna dapat meningkatkan keamanan dan kerahasiaan dengan mengimplementasikan algoritme speck untuk enkripsi dan dekripsi pada QR *Code*
2. Pengguna dapat melakukan penyimpanan informasi secara rahasia (*confidential*) tanpa diketahui orang lain.
3. Pengguna dapat mengimplementasi algoritme SPECK pada QR *Code*
4. Pengguna dapat mengetahui cara validasi *plaintext* dan ciphertext dengan algoritme SPECK.
5. Pengguna dapat mengetahui kinerja pemrosesan enkripsi dan dekripsi algoritme SPECK pada QR *Code*
6. Pengguna dapat menjaga *privacy / confidentiality* informasi untuk menghindari upaya penyadapan, pembajakan, dan hal yang menyebabkan kebocoran dan manipulasi informasi melalui teknik enkripsi dan dekripsi.

1.5 Batasan masalah

Sesuai dengan rumusan masalah diatas, maka batasan masalah yang di lampirkan, yaitu :

1. Tipe algoritme yang digunakan yaitu SPECK 64 bit, dengan *KEY* 128.
2. Dalam batasan masalah pada QR-*Code*, hanya mampu men *scanning* batasan 20 karakter dikarenakan keterbatasan karakter dalam kamera scanning yang hanya mampu meng-*scan* 100 karakter.
3. *KEY* yang digunakan maksimal 16 karakter.

1.6 Sistematika pembahasan

Sistematika penulisan memberikan gambaran dan uraian dari penyusunan skripsi secara garis besar yang meliputi beberapa bab, antara lain:

BAB I PENDAHULUAN

Memuat latar belakang rumusan masalah , tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan, mengenai implementasi SPECK pada QR *Code*,

BAB II LANDASAN KEPUSTAKAAN

Berisi kajian pustaka, referensi, dan sumber-sumber yang berhubungan dengan permasalahan dalam skripsi antara lain mengenai algoritme SPECK, QR *code*, manualisasi SPECK serta teori-teori lainnya sebagai dasar penulisan skripsi.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan metode yang digunakan dalam penelitian, yaitu studi literatur, fungsi algoritme SPECK, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis, dan pengambilan kesimulan dan saran.

BAB IV PERANCANGAN

Pada bab ini dijelaskan analisis dan implementasi algoritme SPECK untuk enkripsi dan dekripsi pada QR *code* yang dapat menjawab permasalahan yang telah diuraikan pada rumusan masalah.

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas tentang implementasi sistem keamanan berupa implementasi algoritme SPECK, implementasi halaman dekripsi, implementasi halaman enkripsi pada *test vector,* implementasi halaman dekripsi pada *test vector* dan pengujian berdasarkan metode penelitian yang telah dibuat yaitu pengujian *test vector,* pengujian fungsional,

pengujian *non* fungsional, dan pengujian kemanan untuk diketahui hasilnya.

BAB VI PENUTUP

Bab ini memuat tentang kesimpulan yang diperoleh dari pembuatan dan pengujian aplikasi yang dikembangkan dalam penelitian skripsi disertai saran mengenai implementasi algoritme SPECK untuk enkripsi dan dekripsi pada QR-*Code*.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan pada bab ini membahas tentang pemilihan dan perbandingan algoritme SPECK. Algoritme Speck adalah salah satu algoritme *Lightweight block cipher* guna memenuhi kebutuhan suatu metode penyandian pada suatu peralatan yang memiliki sumber daya yang sangat terbatas, seperti dalam hal memori, daya komputasi, dan pasokan baterai. Algoritme Speck, yang diajukan oleh peneliti yang bekerja pada *National Security Agency* (NSA) di Amerika.

2.1 Kajian Pustaka

Dalam makalah ini, dibahas mengenai pengujian algoritme Speck menggunakan uji *Strict Avalanche Criterion* (SAC) untuk melihat pola dari *output* yang dihasilkan jika diubah satu bit input-nya. Tujuan uji SAC ini adalah untuk menguji algoritme merupakan pemetaan acak atau bukan. Pemetaan acak adalah persyaratan yang harus dimiliki suatu algoritme *block cipher*. Dalam pengujian ini, diambil sampel acak sebanyak 220 buah dan pengujian dilakukan hanya pada 5 (lima) varian dari algoritme Speck tersebut. Kelima algoritme Speck yang diuji adalah yang memiliki panjang pesan sebanyak 32, 48 dan 64. Dari hasil pengujian tersebut diperoleh kesimpulan bahwa algoritme Speck merupakan pemetaan acak dengan *plaintext* sebagai variabel bebas (Firmanesa, 2016).

Penelitian terdahulu tentang SPECK diantaranya yang diteliti Mohanti (2013) *A Novel SPECK Algorithm for Faster Image Compression Filter* yang diusulkan digunakan melalui dekomposisi dengan penambahan tahap pra-proses untuk algoritme SPECK untuk kompresi citra uji standar. Modifikasi tersebut memberikan PSNR secara signifikan lebih tinggi. Skema pra-pemrosesan yang diusulkan, yang disebut *Zero-Shifting*, membawa nilai DC dalam kisaran integer yang ditandatangani serupa dengan nilai AC, sehingga perhitungan koefisien yang berubah menjadi lebih konsisten. Penelitian tersebut menyimpulkan peningkatan yang signifikan dalam pengkodean dan penguraian waktu dengan membuat *codec* lebih cepat daripada yang asli.

Pada penelitian sebelumnya, (MItra & Rakesh, 2016)meneliti *Desktop Application of QR Code for Data Security and Authentication.* Pada aplikasi ini menggunakan AES sebagai enkripsi. Sebagai kode QR dapat disesuaikan atau diformat sesuai keinginan perusahaan dengan menerapkan warna, embedding logo image atau label, maka akan membantu organisasi untuk mengkomunikasikan informasi dengan cara yang lebih menekankan dan membedakan daripada yang lain. Kode QR yang aman akan membantu perusahaan untuk mengkomunikasikan informasi dengan aman sehingga menghasilkan tujuan ganda pertukaran data yang aman dan tidak jelas dan akan membantu perusahaan menerapkan solusi untuk pertukaran informasi yang

terkompresi, hemat biaya, aman dan tersembunyi. Namun, penelitian tersebut belum menerapkan alagortime SPECK.

Pada penelitian berikutnya, (Sholeh & Muharom, 2016) menyimpulkan bahwa penggunaan smart presensi yang dikombinasikan dengan teknologi QR-*Code* dapat memberikan kepraktisan dan dapat memberikan solusi agar presensi berjalan dengan baik dan efisien. Pengawas ujian tidak lagi membubuhkan tanda tangan pada kartu ujian dan mahasiswa tidak lagi mencetak kartu ujian. Pemanfaatan fungsi dari *smartphone* akan memudahkan dosen dalam melakukan presensi secara *online*. Nomor Ujian dan NIM mahasiswa akan tersimpan dalam database dan akan ditampilkan menggunakan QR-*Code*, saat pengawas ujian melakukan *scanning* QR-*Code*, maka mahasiswa akan menyerahkan QR-*Code* yang telah tercetak di HP. Pengamanan data yang dilakukan adalah dengan memanfaatkan kode batang QR *Code* menggunakan enkripsi *vigenere cipher*. Aplikasi Smart Presensi pada Ujian di Universitas Muhammadiyah Jember merupakan alternative untuk mempermudah dan menyederhanakan proses presensi.

2.2 Sistem dan Sistem Keamanan

Sistem keamanan adalah tindakan pencegahan terhadap segala bentuk penyebab kerugian, termasuk di dalamnya kerugian secara fisik dan *non* fisik, berwujud atau tidak berwujud, serta adanya bermacam-macam kerugian oleh berbagai sebab.

Garfinkel mengemukakan bahwa sistem keamanan melingkupi lima aspek, meliputi (Widiyanto, 2007):

1. *Privacy / Confidentiality*
   Aspek *privacy* atau *confidentiality* adalah sebuah tindakan yang dilakukan untuk menjaga informasi dari orang yang tidak berhak mengakses informasi tersebut. *Privacy* lebih ke arah data-data yang bersifat rahasia sedangkan *confidentiality* berhubungan dengan data yang diberikan kepada pihak lain dengan maksud dan tujuan tertenu.
2. *Integrity*
   Aspek *integrity* atau integritas lebih menekankan bahwa suatu informasi tidak boleh diubah tanpa adanya izin dari pemilik informasi tersebut. Jika terdapat perbedaan maka boleh dibilang aspek integritas tidak tercapai. Adanya virus, *trojan horse*, dan sejenisnya merupakan salah satu hal yang umumnya mengakibatkan pengubahan sebuah informasi.
3. *Authentication*
   Aspek ini berhubungan dengan metode untuk menyatakan bahwa informasi betul-betul asli, orang yang mengakses atau memberikan

informasi adalah betul-betul orang yang dimaksud, atau server yang ditujukan adalah server yang asli.

4. *Availability*
   Aspek *availability* berhubungan dengan ketersediaan sebuah data atau informasi. Data maupun informasi tersebut hanya dapat digunakan oleh yang berhak.
5. *Access Control*
   Aspek *access control* berhubungan dengan cara pengaturan akses kepada informasi. Misalnya, seorang admicoderator memiliki hak akses penuh terhadap sebuah komputer, tetapi hal ini tidak berlaku bagi *account guest* ataupun *limited account* lainnya.

2.3 Algoritme SIMON & SPECK

Simon dan Speck diusulkan secara terbuka pada bulan Juni 2013 oleh sekelompok peneliti di Direktorat Riset Keamanan Nasional AS. Karya desain dimulai pada tahun 2011, dan jumlah kriptanalisis yang signifikan telah dilakukan (tidak hanya oleh para desainer, namun oleh banyak orang di seluruh perusahaan) pada saat menjelang publikasi. Banyak cipher blok ringan telah diusulkan, dan banyak tampil dengan baik di berbagai platform terbatas. Simon dan Speck masing-masing memiliki beberapa instantiasi, yang mendukung ukuran blok 32, 48, 64, 96, dan 128 bit, dan dengan tiga ukuran kunci untuk disesuaikan dengan ukuran masing-masing blok. Setiap keluarga menyediakan sepuluh algoritme dalam semua. Tabel 2.2 mencantumkan blok dan ukuran kunci yang berbeda, dalam bit. Algoritme SPECK adalah salah satu algoritme *block cipher* yang telah dirilis oleh National Security Agency (NSA) pada Juni 2013. Karena *block cipher*, maka algoritme SPECK menggunakan input (masukan) berupa blok (kumpulan bit) bukan bit. SPECK termasuk aplikasi yang ringan dan sangat baik dijalankan di dalam *software*. SPECK dirancang untuk dapat berjalan baik di *software* maupun *hardware*, terutama di dalam *microcontroller*. Algoritme ini memiliki besar kunci dan blok yang berbeda-beda tergantung pada kasus yang dihadapi. Satu blok akan terdiri 2 word. 1 word dapat berukuran 16, 24, 32, 48, atau 64 bit (Beaulieu, R., *et. al*., 2013).

Tabel 2.1 Simon & Speck

Block size	*Key sizes*
32	*64*
48	*72,96*
64	*96,128*
96	*96,144*
128	*128,192,256*

Sumber : (Beaulieu, 2013)

Kesederhanaan desain memiliki keuntungan tambahan, Speck memiliki *throughput* tertinggi pada prosesor 64 bit dari setiap cipher blok yang diimplementasikan dalam perangkat lunak. Sebagian besar algorita kriptografi yang dirancang untuk memenuhi kebutuhan komputasi dekstop pada saat ini. Bidang kriptografi yang lebih sederhana membahas keamanan dari suatu perangkat dengan sangat terbatas. Tujuan dari algoritme jenis ini adalah memberikan fleksibilitas dankarakteristik kinerja yang dibutuhkan pengembang. Algoritme jenis ini menekankan pada cipher blok tidak menyediakannya keamanan dengan sendirinya. Aplikasi yang berbeda juga akan memberikan tingkat keamanan yang berbeda dengan syarat tertentu serta pengembangan protokol yang harus dikembangkan sendiri-sendiri demi mencapai tingkat keamanan yang diinginkan. *Cipher* blok memiliki beberapa kegunaan dalam kriptogtafi primitif dan setiap protokol ringan bisa didasarkan pada cipher dengan ukuran yang tepat. Terdapat varian dari algoritme Speck berdasarkan panjang kunci dan ukuran blok pesannya seperti dapat dilihat pada Tabel 2.3 dan Tabel 1 Berikut notasi yang digunakan pada algoritme Speck (Beaulieu, 2013):

Tabel 2.2 Notasi Algoritme Speck

⊕	Operator bitwise *XOR*
n	Ukuran panjang word pada SPECK (*n* = 16,24,32,48, atau 64)
+	Operator penjumlahan modulo 2^n
-	Operator pengurangan modulo 2^n
<<j	Rotasi bit ke kiri sebanyak *j* bit
>>j	Rotasi bit ke kanan sebanyak *j* bit
R	Jumlah *round*
a→b	Memperbarui nilai *a* dengan nilai *b*

Sumber : (Firmanesa, 2016)

Tabel 2.3 Varian algoritme Speck dan parameter yang digunakan

Sumber : (Firmanesa, 2016)

Block size $2n$	Key size mn	Word size n	Key words m	Rot α	Rot β	Rounds r
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3	8	3	26
	128		4			27
96	96	48	2	8	3	28
	144		3			29
128	128	64	2	8	3	32
	192		3			33
	256		4			34

Besar blok dan kunci akan menentukan seberapa banyak ronde yang akan dipakai, rinciannya dapat dilihat pada Tabel 2.5.

Tabel 2.5 Versi Algoritme SPECK

Block Size (bits)	***Key Size (bits)***	***Rotation α***	***Rotation β***	***Rounds***
2 x 16 = 32	4 x 16 = 64	7	2	22
2 x 24 = 48	3 x 24 = 72	8	3	22
	4 x 24 = 96			23
2 x 32 = 64	3 x 32 = 96	8	3	26
	4 x 32 = 128			27
2 x 48 = 96	2 x 48 = 96	8	3	28
	3 x 48 = 144			29
2 x 64 = 128	2 x 64 = 128	8	3	32
	3 x 64 = 192			33
	4 x 64 = 256			34

Sumber **:** (Ray Beaulieu et al, 2013)

2.3.1 Manualisasi SPECK

Berikut langkah – langkah untuk membuat manualisasi pada algoritme speck:

$X_0 \leftarrow \mathtt{LSL}(X_0)$ (logical shift left)
$X_1 \leftarrow \mathtt{ROL}(X_1)$ (rotate left through carry)
$X_2 \leftarrow \mathtt{ROL}(X_2)$ (rotate left through carry)
$X_3 \leftarrow \mathtt{ROL}(X_3)$ (rotate left through carry)
$X_0 \leftarrow \mathtt{ADC}(X_0, Z_0)$ (add with carry)

Gambar 2.1 Rotasi 1 bit dari 32 bit

Sumber : (Beaulieu, 2013)

Mnemonic	Operation	Register Contents	Cycles
Load	$K \leftarrow k$	$K = k$	12
Add	$X \leftarrow S^8(S^{-8}(X) + Y)$	$X = S^8(S^{-8}(x) + y)$	4
XOR	$K \leftarrow K \oplus S^{-8}(X)$	$K = (S^{-8}(x) + y) \oplus k$	4
Rotate	$Y \leftarrow S^3(Y)$	$Y = S^3(y)$	15
XOR	$Y \leftarrow Y \oplus K$	$Y = S^3(y) \oplus (S^{-8}(x) + y) \oplus k$	4
Move	$X \leftarrow K$	$X = (S^{-8}(x) + y) \oplus k$	2

Gambar 2.2 Implementasi speck round function

Sumber : (Beaulieu, 2013)

Mnemonic	Operation	Register Contents	Cycles
Load	$K \leftarrow k$	$K = k$	12
Add	$X \leftarrow S^8(S^{-8}(X) + Y)$	$X = S^8(S^{-8}(x) + y)$	4
XOR	$K \leftarrow K \oplus S^{-8}(X)$	$K = (S^{-8}(x) + y) \oplus k = x_1$	4
Rotate	$Y \leftarrow S^3(Y)$	$Y = S^3(y)$	15
XOR	$Y \leftarrow Y \oplus K$	$Y = S^3(y) \oplus (S^{-8}(x) + y) \oplus k = y_1$	4
Load	$X \leftarrow l$	$X = l$	12
Add	$K \leftarrow S^8(S^{-8}(K) + Y)$	$K = S^8(S^{-8}(x_1) + y_1)$	4
XOR	$X \leftarrow X \oplus S^{-8}(K)$	$X = (S^{-8}(x_1) + y_1) \oplus l$	4
Rotate	$Y \leftarrow S^3(Y)$	$Y = S^3(y_1)$	15
XOR	$Y \leftarrow Y \oplus X$	$Y = S^3(y_1) \oplus (S^{-8}(x_1) + y_1) \oplus l$	4

Gambar 2.3 Implementasi speck 2 round function

Sumber : (Beaulieu, 2013)

Mnemonic	Operation	Register Contents	Cycles
Load	$K \leftarrow k$	$K = k$	8
Add	$X \leftarrow S^8(S^{-8}(X) + Y)$	$X = S^8(S^{-8}(x) + y)$	4
XOR	$K \leftarrow S^{-8}(X) \oplus K$	$K = (S^{-8}(x) + y) \oplus k = x_1$	4
Rotate	$X \leftarrow S^3(Y)$	$X = S^3(y)$	14
XOR	$X \leftarrow X \oplus K$	$X = S^3(y) \oplus (S^{-8}(x) + y) \oplus k = y_1$	4
Load	$Y \leftarrow l$	$Y = l$	8
Add	$K \leftarrow S^8(S^{-8}(K) + X)$	$K = S^8(S^{-8}(x_1) + y_1)$	4
XOR	$Y \leftarrow S^{-8}(K) \oplus Y$	$Y = (S^{-8}(x_1) + y_1) \oplus l = x_2$	4
Rotate	$K \leftarrow S^3(X)$	$K = S^3(y_1)$	14
XOR	$K \leftarrow K \oplus Y$	$K = S^3(y_1) \oplus (S^{-8}(x_1) + y_1) \oplus l = y_2$	4
Load	$X \leftarrow m$	$X = m$	8
Add	$Y \leftarrow S^8(S^{-8}(Y) + K)$	$Y = S^8(S^{-8}(x_2) + y_2)$	4
XOR	$X \leftarrow S^{-8}(Y) \oplus X$	$X = (S^{-8}(x_2) + y_2) \oplus m = x_3$	4
Rotate	$Y \leftarrow S^3(K)$	$Y = S^3(y_2)$	14
XOR	$Y \leftarrow Y \oplus X$	$Y = S^3(y_2) \oplus (S^{-8}(x_2) + y_2) \oplus m = y_3$	4

Gambar 2.4 Implementasi speck 3 round function

Sumber : (Beaulieu, 2013)

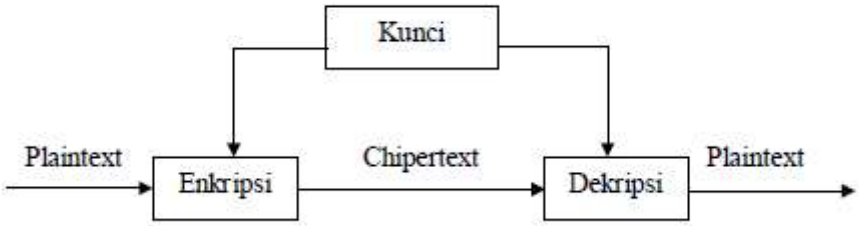
2.4 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani: "*cryptos*" dan "*graphein*". *Crypto* berarti rahasia (*secret*), sedangkan *graphein* berarti tulisan (*writing*). Sehingga, kriptografi berarti "*secret writing*" (tulisan rahasia). Secara terminologi, kriptografi merupakan ilmu dan seni yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentifikasi data. Sebuah algoritme kriptografi, disebut *cipher*, merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi. Biasanya kedua persamaan matematik (enkripsi dan dekripsi) tersebut memiliki hubungan matematis yang cukup erat.

Kriptografi sendiri mempunyai komponen-komponen untuk mencapai tujuan kriptografi.Menurut (Ariyus, 2006), 2009 beberapa komponen dalam kriptografi meliputi:

1. Enkripsi (*Encryption*)
   Enkripsi merupakan hal yang sangat penting dalam kriptografi untuk mengamankan sebuah informasi agar pesan yang dikirimkan terjaga kerahasiaannya. Informasi (yang disebut *plaintext*) diubah menjadi serangkaian kode rumit yang sulit diartikan. Enkripsi sendiri bisa diartikan sebagai *cipher* atau kode. Berdasarkan ISO 7498-2, terminologi yang lebih tepat digunakan untuk menamakan proses ini adalah "*encipher*".
2. Dekripsi (*Decryption*)
   Dekripsi merupakan kebalikan dari proses enkripsi. Dekripsi yaitu proses mengubah kembali pesan yang telah dienkripsi menjadi pesan aslinya, yang disebut dengan dekripsi pesan. Berdasarkan ISO 7498-2, terminologi yang lebih tepat untuk menamakan proses ini adalah "*decipher*".
3. Kunci
   Kunci yang dimaksud disini adalah kunci atau sandi yang digunakan untuk melakukan enkripsi maupun dekripsi. Kunci terbagi menjadi dua bagian, yaitu kunci privat (*private key*) dan kunci publik (*public key*).
4. *Plaintext*
   Plaintext disebut juga *cleartext*, yaitu pesan asli yang ditulis atau diketik. *Plaintext* inilah yang akan diproses menggunakan algoritme kriptografi agar menjadi *ciphertext*.
5. Pesan
   Pesan bisa berupa data atau informasi yang dikirimkan (melalui kurir, saluran komunikasi data, dan sebagainya) atau yang disimpan di dalam media penyimpanan (kertas, *storage*, dan sebagainya).
6. *Ciphertext*
   Ciphertext merupakan pesan yang dihasilkan dari proses enkripsi. Pesan yang terkandung dalam *ciphertext* ini sulit dibaca karena berisi berbagai macam karakter tanpa arti/tidak bermakna.
7. Kriptanalisis (*Cryptanalysis*)
   Dapat diartikan sebagai analisis sandi atau suatu ilmu memecahkan *ciphertext* menjadi *plaintext* tanpa mengetahui kunci yang digunakan. Pelakunya disebut *cryptanalys* (kriptanalis).

Kriptografi mempunyai dua komponen utama yaitu enkripsi dan dekripsi. Selain itu dibutuhkan kunci untuk mengubah *plaintext* menjadi *ciphertext*, begitu juga sebaliknya. Tanpa kunci, *plaintext* tidak bisa melakukan enkripsi pesan menjadi *ciphertext*, juga sebaliknya. Kerahasiaan kunci ini sangatlah penting, apabila kerahasiaannya terbongkar maka isi pesan akan terbongkar. Berikut adalah skema ilustrasi proses enkripsi dan dekripsi.



Gambar 2.5 Gambar Skema Proses Enkripsi dan Dekripsi

Sumber: (Munir, 2006)

Pada gambar 2.1 mengilustrasikan sebuah pesan/*plaintext* dienkripsi menggunakan kunci enkripsi sehingga menjadi *ciphertext* yang akan didekripsi menggunakan kunci dekripsi untuk menghasilkan *plaintext* kembali. Klasifikasi Kriptografi, Dari sisi era perkembangannya, ilmu kriptografi dibagi menjadi dua, yaitu kriptografi klasik dan kriptografi modern.

2.4.1 Kriptografi Klasik

Kriptografi klasik merupakan awal dari perkembangan seni ataupun ilmu kriptografi. Pada era perkembangan ini kekuatan kriptografi terletak pada kerahasiaan algoritme yang digunakan, jenis algoritme ini dinamakan algoritme *restricted* (algoritme terikat). Namun di tengah perkembangannya algoritme *restricted* ini ditemukan banyak kelemahan sehingga kurang mampu menyesuaikan keamanan data dan informasi saat ini. Adapun kelemahan-kelemahan algoritme *restricted* adalah sebagai berikut:

- Algoritme *restricted* merupakan algoritme bersifat rahasia, sehingga kemampuan algoritme tidak pernah diuji oleh para pakar kriptografi dan berpengaruh pada ketidakpercayaan *user* terhadap kemampuannya.
- Apabila terjadi kebocoran rahasia algoritme, maka harus dikembangkan lagi algoritme baru sebagai penggantinya. Konsekuensinya adalah pemborosan biaya karena pembuatan algoritme kriptografi baru yang cukup mahal.

2.4.2 Kriptografi Modern

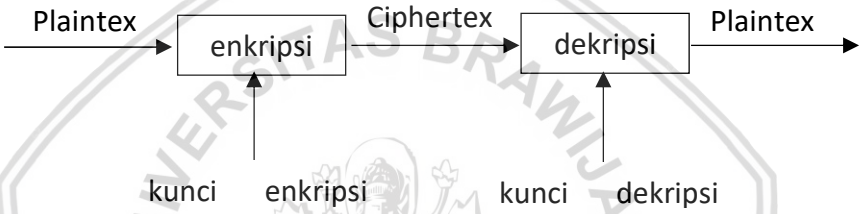
Kriptografi modern dikembangkan untuk memperbaiki kelemahan-kelemahan yang ada dalam kriptografi klasik. Berlawanan dengan kriptografi klasik, algoritme yang digunakan pada kriptografi modern dapat diketahui oleh publik sehingga tidak bersifat rahasia. Hal ini dilakukan untuk menutupi kelemahan kriptografi klasik, sehingga algoritme dari kriptografi modern dapat diukur ketangguhannya oleh pakar-pakar kriptografi. Kekuatan kriptografi ini bertumpu pada kerahasiaan kunci penyandiannya.

2.4.3 Algoritme Kriptografi

Algoritme kriptografi disini termasuk dalam kriptografi modern karena menggunakan kunci sandi sebagai tolak ukur kemampuannya. Berdasarkan kunci penyandiannya, algoritme kriptografi dibagi menjadi tiga jenis yaitu:

2.4.4 Algoritme Simetris

Algoritme simetris (*Symnetric Algorithm*) merupakan suatu algoritme dimana kegiatan enkripsi maupun dekripsi menggunakan satu kunci yang sama sehingga algoritme ini disebut juga sebagai *single-key algorithm*. Keamanan dari informasi yang menggunakan algoritme kunci simetris tergantung pada kunci. Sebelum pengiriman informasi dilakukan, baik pengirim dan penerima harus memilih suatu kunci tertentu untuk digunakan bersama, kunci ini haruslah rahasia dari pihak yang tidak berkepentingan. Hal ini juga yang membuat algoritme simetris disebut *secret/private-key algorithm*.



Gambar 2.6 Diagram Proses Enkripsi dan Dekripsi Algoritme Simetris

Sumber: (Munir, 2006)

Pada Gambar 2.2 merupakan diagram proses enkripsi dan dekripsi pada algoritme simetris. Proses berawal dari *plaintext* yang dienkripsi menggunakan K sebagai kunci enkripsi yang menghasilkan *ciphertext.* Selanjutnya *ciphertext* di dekripsi kembali dengan kunci yang sama (K) untuk menghasilkan *plaintext.*

Kelebihan algoritme simetris:

- Kecepatan operasi yang lebih tinggi dibandingkan algoritme asimetris.
- Ukuran kunci relatif lebih pendek.
- Karena kecepatannya yang cukup tinggi, maka dapat digunakan pada sistem *real-time*.

Kelemahan algoritme simetris:

- Untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.
- Kunci harus sering diubah, terutama pada setiap sesi komunikasi.
- Kunci harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasiaan kunci ini. Kasus dalam pengiriman kunci ini disebut "*key distribution problem*".

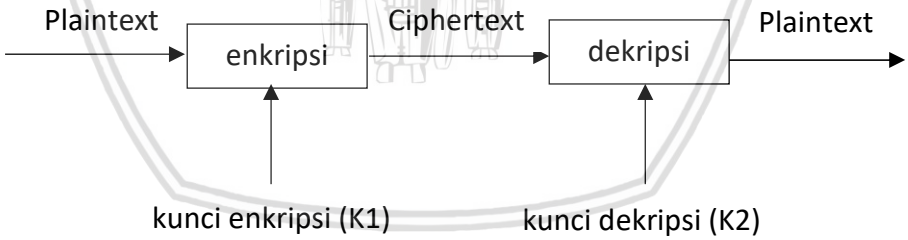
Beberapa contoh algoritme yang menggunakan kunci simetris yaitu *DES (Data Encryption Standard), Blowfish, Triple-DES, IDEA, Serpent, On Time Pad (OTP) dan AES (Advanced Encryption Standard).*

Algoritme kriptografi (*cipher*) simetris dapat dikelompokkan menjadi dua berdasarkan jumlah data tiap proses dan alur pengolahan data di dalamnya:

1. *Stream Cipher* (Cipher Aliran)
   Stream Cipher merupakan algoritme kriptografi yang beroperasi pada *plaintext* atau *ciphertext* dalam bentuk *bit* tunggal, yang dalam hal ini rangkaian *bit* dienkripsikan atau didekripsikan pada tiap *bit*. Yang termasuk dalam kelompok *cipher* aliran adalah algoritme vernam cipher.
2. *Block Cipher* (Cipher Blok)
   Block Cipher merupakan algoritme kriptografi yang beroperasi pada *plaintext* atau *ciphertext* dalam bentuk blok *bit*, yang dalam hal ini rangkaian *bit* dibagi menjadi blok-blok *bit*, yang panjangnya sudah ditentukan sebelumnya. Jika panjang blok adalah 64 *bit*, maka algoritme enkripsi memperlakukan 8 karakter setiap kali penyandian (1 karakter = 8 *bit* dalam pengkodean ASCII).

2.4.5 Algoritme Asimetris

Berbeda dengan algoritme kunci simetris, kunci yang digunakan pada algoritme asimetris dibuat sepasang, satu kunci untuk enkripsi dan satu kunci untuk dekripsi. Kunci untuk enkripsi dapat diketahui oleh siapapun dan secara bebas didistribusikan umum, sehingga dinamakan kunci publik. Sedangkan kunci untuk dekripsi bersifat rahasia sehingga dinamakan kunci privat. Seperti yang dijelaskan pada Gambar 2.3:



Gambar 2.7 Diagram Proses Enkripsi dan Dekripsi Algoritme Asimetris

Sumber: (Munir, 2006)

Pada gambar 2.3 merupakan diagram proses enkripsi dan dekripsi pada algoritme asimetris. Proses berawal dari *plaintext* yang dienkripsi menggunakan K1 sebagai kunci enkripsi dan menghasilkan *ciphertext*. Selanjutnya *ciphertext* dilakukan dekripsi menggunakan K2 sebagai kunci dekripsi untuk kembali menampilkan *plaintext.* Sehingga kunci yang digunakan untuk proses enkripsi maupun dekripsi menggunakan kunci yang berbeda.

Kelebihan algoritme asimetris:

- Sangat jarang dilakukan perubahan pada *public key* dan *private key*
- Masalah keamanan pada distribusi kunci lebih baik
- Masalah manajemen kunci yang lebih baik karena jumlah kunci yang lebih sedikit

Kelemahan algoritme asimetris:

- Kecepatan yang lebih rendah dibandingkan algoritme simetris
- Untuk tingkat keamanan yang sama, kunci yang digunakan lebih panjang dibandingkan dengan algoritme simetris (ukurannya lebih besar dibandingkan algoritme simetris)
- Tidak adanya jaminan bahwa *public key* benar-benar aman

2.4.6 Fungsi Hash

Fungsi Hash sering disebut dengan fungsi hash satu arah (*one-way function*), *message digest*, *fingerprint*, fungsi kompresi, dan *message authentication code* (MAC), merupakan fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap.

Keluaran fungsi hash disebut dengan nilai hash (hash value). Fungsihash digunakan untuk memerika keaslian sebuah salinan arsip. Salinan arsip ini diuji keasliannya dengan cara membandingkan nilai hash-nya dengan nilai hash arsip yang asli yang sudah diketahui (Lanelo & A, 2011). Fungsi hash digunakan karena setiap bit yang dihasilkan oleh fungsi hash bergantung kepada setiap bit pada pesan asli (Fahmi, 2007).

Sebuah masukan dalam fungsi hash dibagi ke dalam blok-blok yang memiliki ukuran yang sama. Fungsi hash digunakan kepada setiap blok pesan dengan memasukkan blok pesan dan hasil blok pesan sebelumnya seperti yang ditunjukkan pada Gambar 2.4. Dimana *hi* merupakan keluaran dari fungsi hash untuk blok ke-i dan *Mi* menyatakan blok pesan ke-i.

2.5 Ilustrasi Key Ekspansi, Enkripsi dan Dekripsi pada Algoritme SPECK

Berikut merupakan ilustrasi key ekspanasi, enkripsi dan dekripsi pada algoritma yang terdiri dari key ekspansi, enkripsi dan dekripsi

2.5.1 Pembangkitan Kunci pada SPECK

Proses pembangkitan kunci pada algoritme Speck menggunakan strutur yang sama dengan fungsi enkripsinya, tetapi kunci yang digunakan diganti menjadi nilai *round,* pembangkitan kunci bertujuan untuk ekspansi (perpanjangan) kunci disini terdapat elemen (pedding), yang akan membuat kunci semakin panjang

dari elemen 4 menjadi elemen 25, masing2 elemen berukuran 32 bit (4 karakter). Buat putaran sebanyak T-2 putaran. Buat pergeseran key awal (digeser ke kanan sebanyak alfa). Selanjutnya buat key ekspansi (diegeser ke kiri sebanyak Beta) . Berikut proses pembangkitan kunci pada algoritme Speck :

Proses Pembangkitan Kunci Algoritma Speck

Input : $K_{(mn)}$

Output : $k_0, \dots, k_{r-1}$

Proses :

1. $k_{0(n)}|l_{0(n)}| \dots |l_{m-2(n)} \leftarrow K_{(mn)}$
2. Untuk $i = 1$ sampai dengan $r - 1$
   $tmp \leftarrow ((l_0 \gg \alpha) + k_i) \oplus i$
   $k_{i+1} \leftarrow (k_i \ll \beta) \oplus tmp$
   Untuk $j = 0$ sampai dengan $m - 1$
   $l_j \leftarrow l_{j+1}$
   $l_{m-2} \leftarrow tmp$
3. *Output* :
   $k_{0(n)}| k_{1(n)}| \dots |k_{r-1(n)}$

Gambar 2.8 Pembangkitan kunci pada speck

Sumber : (Firmanesa, 2016)

Key schedule atau pembangkitan kunci pada algoritme SPECK digunakan untuk membuat kunci setiap ronde. Kunci yang telah dibuat akan digunakan di fungsi ronde setiap ronde. Pada proses ini terdapat 2 variabel yang akan diproses, yaitu k_i an l_i.

Untuk mendapatkan nilai l_i dilakukan rumus sebagai berikut

$$l_{i+m-1} = (k_i + S^{-\alpha} l_i) \oplus \text{i} \quad (2.1)$$

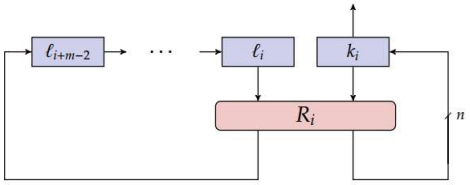
Pada rumus 2.1, nilai m adalah nilai *word* pada *key* (tergantung jenis SPECK yang digunakan). Nilai i adalah ronde saat ini. Nilai i akan terus bertambah dari 0 sampai jumlah ronde sesuai jenis SPECK yang digunakan. Nilai l_{i+m-1} didapatkan dengan cara melakukan *addition modulo* 2^n antara k_i dengan l_i yang telah dilakukan rotasi ke kanan sejumlah α bit. Kemudian hasilnya akan dilakukan operasi XOR dengan nilai i.

Untuk mendapatkan nilai k_i, dilakukan rumus sebagai berikut

$$k_{i+1} = S^{\beta} k_i \oplus l_{i+m-1} \quad (2.2)$$

Pada rumus 2.2, nilai k_{i+1} didapatkan dengan cara melakukan operasi XOR antara k_i yang telah dilakukan rotasi ke kiri sejumlah β bit dengan nilai l_{i+m-1} yang didapat dari rumus 2.1.

Kedua rumus ini akan dilakukan perulangan jumlah ronde dari jenis SPECK yang digunakan. Ilustrasi gambaran proses *key schedule* pada algoritme SPECK dapat dilihat pada Gambar 2.5.



Gambar 2.13 Ilustrasi *Key Schedule* pada Algoritme SPECK
Sumber : (Ray Beaulieu et al, 2013)

2.5.2 Enkripsi dari SPECK

Enkripsi dari SPECK bertujuan untuk membuat proses enkripsi, dengan menggunakan kunci yang sudah siterapkan pada key ekspansi. Bertujuan untuk mengembalikan dari bentuk *ciphertext* menjadi plaintext kembali, dengan menggunakan kunci yang sudah di terapkan. Buat putaran sebanyak T-1. Olah nilai X, geser X ke kanan sebanyak alfa. Olah nilai y , geser y ke kiri sebanyak beta

Algoritme enkripsi pada speck adalah sebagai berikut:

Proses Enkripsi Algoritma Speck
Input : $X_{(2n)}; k_0, \dots, k_{r-1}$
Output : $Y_{(2n)}$
Proses :
1. $X_{0(n)}|X_{1(n)} \leftarrow X_{(2n)}$
2. Untuk $i = 1$ sampai dengan $r-1$
   $X_1 \leftarrow ((X_1 \gg \alpha) + X_0) \oplus k_i$
   $X_0 \leftarrow (X_0 \ll \beta) \oplus X_1$
3. *Output* :
   $Y_{(2n)} \leftarrow X_{0(n)}|X_{1(n)}$

Gambar 2.9 Enkripsi pada speck

Sumber : (Firmanesa, 2016)

2.5.3 Dekripsi pada SPECK

Proses untuk melakukan dekripsi pada algoritme memerlukan invers operasi modulo, proses putaran sama dengan enkripsi, perbedaannya pelibatan key ekspansi dalam prosesnya dibalik dari indek ke 26 sampai ke 0 . Berikut proses dekripsi pada algoritme Speck :

Proses Dekripsi Algoritma Speck
Input : $Y_{(2n)}; k_0, \dots, k_{r-1}$
Output : $X_{(2n)}$
Proses :
1. $Y_{0(n)}|Y_{1(n)} \leftarrow Y_{(2n)}$
2. Untuk $i = 1$ sampai dengan $r-1$
   $Y_0 \leftarrow (Y_0 \oplus Y_1) \gg \beta$
   $Y_1 \leftarrow ((Y_1 \oplus k_{r-i-1}) - Y_0) \ll \alpha$
3. *Output* :
   $X_{(2n)} \leftarrow Y_{0(n)}|Y_{1(n)}$

Gambar 2.10 Dekripsi pada SPECK
Sumber : (Firmanesa, 2016)

2.5.4 Fungsi *Round*

Operasi yang dilakukan dalam fungsi ronde ada 3, yaitu operasi *addition modulo* 2^n, rotasi ke kiri atau ke kanan sejumlah bit tertentu, dan XOR. Operasi *addition modulo* 2^n sama seperti melakukan operasi AND atau pertambahan per bit (Beaulieu, R., *et. al*., 2013). Pada fungsi ronde ini terdapat rumus enkripsi dan dekripsi yang dipakai pada algoritme SPECK.

Proses enkripsi dilakukan dengan rumus berikut

$$R_k(x,y) = ((S^{-\alpha}x + y) \oplus \mathrm{k}\ ,\ S^{\beta}y \oplus ((S^{-\alpha}\mathrm{x} + \mathrm{y}) \oplus \mathrm{k})) \qquad (2.3)$$

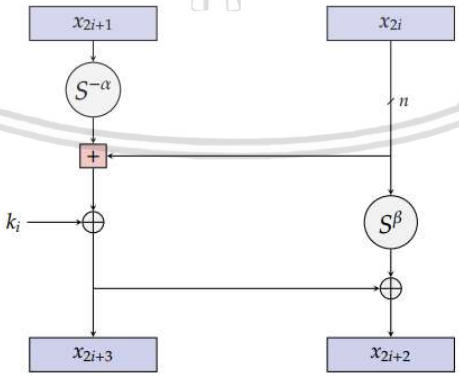
Pada rumus 2.3, x adalah *plaintext* pertama dan y adalah *plaintext* kedua. X didapatkan dari melakukan *addition modulo* 2^n antara hasil x yang dilakukan rotasi ke kanan sejumlah α bit dan y. Kemudian hasil tersebut dilakukan operasi XOR terhadap *k* (key). Sedangkan Y didapatkan dari melakukan operasi rotasi ke kiri sejumlah β bit pada y. Kemudian hasil tersebut dilakukan operasi XOR terhadap x yang telah dimasukan rumus 2.3.

Proses dekripsi dilakukan dengan rumus berikut

$${R_k}^{-1}(x,y) = (S^{\alpha}((x \oplus \mathrm{k}) - S^{-\beta}(x \oplus y))\ ,\ S^{-\beta}\ (\mathrm{x} \oplus \mathrm{y})) \qquad (2.4)$$

Pada rumus 2.4, x didapatkan dengan melakukan operasi XOR pada x dengan *k* (key). Hasil tersebut akan dilakukan rotasi ke kiri sejumalah α bit. Kemudian akan dikurangkan dengan hasil dari y yang sudah diproses pada rumus 2.4.Sedangkan y didapatkan dari melakukan operasi XOR antara x dan y. Kemudian hasilnya akan dirotasi ke kanan sejumlah β bit.

Kedua proses ini akan diulang sejumlah ronde sesuai pada jenis SPECK yang digunakan. Ilustrasi gambaran fungsi ronde pada SPECK dapat dilihat pada Gambar 2.6.



Gambar 2.11 Ilustrasi fungsi Ronde pada Algoritme SPECK
Sumber : (Ray Beaulieu et al, 2013)

2.6 Bahasa Pemrograman PHP

Hypertext preprocessor (PHP) adalah teknologi *server-side scripting* yang digunakan untuk aplikasi *web* yang dinamis dan intraktif (Binus, 2017). Sebuah halaman Hypertext *Preprocessor* (PHP) adalah sebuah halaman *Hypertext Markup Language* (HTML) yang memiliki *server-side scripts* yang ditempatkan dalam server dan diproses *webserver* sebelum dikirim ke *browser* pemakai. *Serverside scripts* dijalankan ketika *browser* melakukan permintaan *file*. *Php* dari *server.* PHP dipanggil oleh *webserver,* dimana proses *script* perintah yang ada disuatu halaman dieksekusi mulai dari awal sampai akhir didalam mesin Hypertext *Preprocessor* (PHP). Setelah *script* PHP diolah, hasilnya akan ditampilkan kepada *client* melalui *web browser* berupa tampilan HTML.

Kelebihan – kelebihan dari Hypertext preprocessor(PHP) yaitu:

- Kecepatan akses yang tinggi
- Dapat bekerja dalam *web server* yang berbeda dan sistem operasi yang berbeda
- PHP adalah *freeware* dan *open source*
- Merupakan bahasa pemrograman yang *embedded*
- Dapat berjalan pada berbagai *platform*: Apache,IIS,Microsoft Personal Web Server

Alasan mengapa menggunakan PHP dan berbasis *web* adalah:

1.Bahasa pemrograman PHP terbukti sangat handal dalam membangun sebuah program berbasis *web.*
2.Waktu yang digunakan untuk memproses data dan menjalankan perintah–perintah *query* sangat cepat.
3.Dengan berjalan dalam sebuah *webserver,*makasecara otomatis program ini bersifat *multiuser*
4.*Database* MySQL menyimpan data didalam direktori khusus yang terpisah dari file program PHP sehingga keamanan data lebih terjamin.
5.*Web server* dan data *base server* terpisah sehingga menyulitkan pihak luar untuk mengakses data yang terdapat di dalam *database.*
6.Bahasa pemograman PHP dan *database* MySQL lebih fleksibel karena dapat diakses oleh sistemoperasi Windows maupun Linux.
7. Program dapat diakses dari komputer manapun tanpa harus mengistal program *client.* Program bantuan untuk mengakses sistem ini hanyalah sebuah browser.

2.7 Quick Respon Code

Quick Response Code sering di sebut *QR Code* atau Kode *QR* adalah semacam simbol dua dimensi yang dikembangkan oleh Denso Wave yang merupakan anak perusahaan dari Toyota sebuah perusahaan Jepang pada tahun 1994. Tujuan dari *QR Code* ini adalah untuk menyampaikan informasi secara cepat dan juga

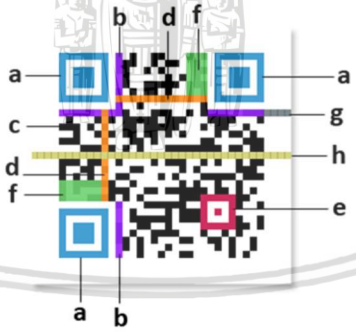
mendapat tanggapan secara cepat. Pada awalnya *QR Code* digunakan untuk pelacakan bagian kendaraan untuk *manufacturing*. Namun sekarang, telah digunakan untuk komersil yang ditujukan pada pengguna telepon seluler. *QR Code* adalah perkembangan dari *barcode* atau kode batang yang hanya mampu menyimpan informasi secara *horizontal* sedangkan *QR Code* mampu menyimpan informasi lebih banyak, baik secara *horizontal* maupun vertikal.

Gambar 2.12 Contoh *QR Code*

QR Code biasanya berbentuk persegi putih kecil dengan bentuk geometris

hitam (dapat dilihat di gambar 2.13), meskipun sekarang banyak yang telah berwarna dan digunakan sebagai *brand* produk. Informasi yang dikodekan dalam *QR Code* dapat berupa *URL*, nomor telepon, pesan *SMS*, *V-Card*, atau teks apapun (Ashford, 2010). *QR Code* telah mendapatkan standarisasi internasional ISO/IEC18004 dan Jepang JIS-X-0510 (Denso, 2011).

2.7.1 Anatomi QR *Code*



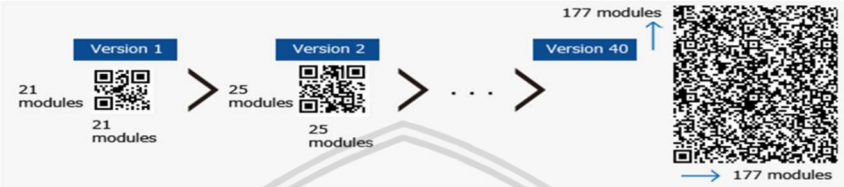
Gambar 2.13 Anatomi *QR Code*

Beberapa penjelasan anatomi *QR Code* Menurut (Ariadi, 2011)antara lain :

a. *Finder Pattern* berfungsi untuk identifikasi letak *QR Code*.
b. *Format Information* berfungsi untuk informasi tentang *error correction level* dan*mask pattern*.
c. *Data* berfungsi untuk menyimpan data yang dikodekan.
d. *Timing Pattern* merupakan pola yang berfungsi untuk identifikasi koordinat pusat*QR Code*, berbentuk modul hitam putih.

e. *Alignment Pattern* merupakan pola yang berfungsi memperbaiki penyimpangan*QR Code* terutama distorsi non linier.
f. *Version Information* adalah versi dari sebuah *QR Code*.
g. *Quiet Zone* merupakan daerah kosong di bagian terluar *QR Code* yangmempermudah mengenali pengenalan *QR* oleh sensor *CCD*.
h. *QR Code version* adalah versi dari *QR Code* yang digunakan.

2.7.2 Versi QR *Code*



Gambar 2.14 Versi QR *Code*

(Sumber: (qrcode.com, 2013)**)**

QR Code dapat menghasilkan 40 versi yang berbeda dari versi 1 (21 x 21 modul) sampai versi 40 (177 x 177 modul). Tingkatan Versi *QR Code* 1 dan 2 berbeda 4 modul berlaku sampai dengan versi 40. Setiap versi memiliki konfigurasi atau jumlah modul yang berbeda. Modul ini mengacu pada titik hitam dan putih yang membentuk suatu *QR Code*. Setiap versi *QR Code* memiliki kapasitas maksimum data, jenis karakter dan tingkat koreksi kesalahan. Jika Jumlah data yang ditampung banyak maka modul yang yang akan diperlukan dan menjadikan *QR Code* menjadi lebih besar (Denso, 2011).

2.7.3 Mengoreksi Kesalahan QR *Code*

QR Code mampu mengoreksi kesalahan dan pengembalian data dalam pembacaan kode apabila *QR code* kotor atau rusak. Menurut (Denso, 2011), Ada 4 tingkatankoreksi kesalahan dalam *QR code*:

Tabel 2 Level Koreksi

Level Koreksi kesalahan Jumlah	Jumlah Perkiraan Koreksi
L	7%
M	15%
Q	25%
H	30%

Semakin tinggi tingkat koreksi kesalahan semakin besar juga versi QR *Code*. Faktor lokasi dan lingkungan operasi perlu di timbangkan dalam menentukan level *QR Code*.Level Q dan H baik digunakan di pabrik yang kotor, sedangkan L untuk tempat yang bersih. Level yang sering digunakan adalah level M dengan perkiraan koreksi mencapai 15% (qrcode.com, 2013).

2.7.4 Manfaat QR *Code*

Beberapa manfaat yang terdapat pada *QR Code* menurut (Denso, 2011)antara lain:

1. Kapasitas tinggi dalam menyimpan data
   Sebuah *QR Code* tunggal dapat menyimpan sampai 7.089 angka.
2. Ukuran yang kecil
   Sebuah *QR Code* dapat menyimpan jumlah data yang sama dengan *barcode 1D* dan tidak memerlukan ruang besar.
3. Dapat mengoreksi kesalahan
   Tergantung pada tingkat koreksi kesalahan yang dipilih, data pada *QR code* yang kotor atau rusak sampai 30% dapat diterjemahkan dengan baik.
4. Banyak jenis data
   QR Code dapat menangani angka, abjad, simbol, karakter bahasa Jepang, Cina atau Korea dan data biner.
5. Kompensasi distorsi
   QR Code tetap dapat dibaca pada permukaan melengkung atau terdistorsi.
6. Kemampuan menghubungkan
   Sebuah *QR Code* dapat dibagi hingga 16 simbol yang lebih kecil agar sesuai dengan ruang. Simbol-simbol kecil yang dibaca sebagai kode tunggal apabila di *scan* menurut urutan.

2.7.5 Macam-Macam QR *Code*

a) *Qr code* model 1 dan model 2

   i. *Qr code* model 1

Gambar 2.15 Contoh *QR Code* Model 1

Sumber **:** (qrcode.com, 2013)

Model 1 adalah *QR Code* asli, dapat menampung 1.167 angka dengan versi maksimum 14 (73 x 73 modul) (qrcode.com, 2013).

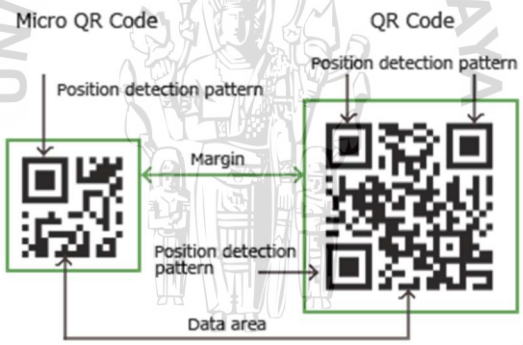
ii. Qr code model 2

Gambar 2.16 Contoh *QR Code* Model 2

Sumber:(qrcode.com, 2013)

Model 2 adalah penyempurnaan dari model 1 dengan versi terbesar 40 (177 x 177 modules), yang mampu menyimpan sampai 7.089 angka (qrcode.com, 2013).

b) Micro qr code



Gambar 2.17 Contoh *Micro QR Code*

Sumber :(qrcode.com, 2013)

Versi terbesar dari kode ini adalah M4 (17 x 17 modul) yang dapat menyimpan hingga 35 angka. Fitur utama dari *Micro QR Code* adalah hanya memiliki satu pola deteksi posisi, dibandingkan dengan regular *QR Code* yang memerlukan sejumlah tempat karena pola deteksi posisi yang terletak di tiga sudut simbol. *QR Code* biasa membutuhkan setidaknya empat modul yang lebar di sekitar simbol, sedangkan *Micro QR Code* hanya membutuhkan cukup dua modul margin. Konfigurasi *Micro QR Code* memungkinkan pencetakan di tempat lebih kecil dari *QR Code* (qrcode.com, 2013).

c) iQR Code

Gambar 2.18 Contoh *iQR Code*

Sumber:(qrcode.com, 2013)

Kode yang dapat dihasilkan dari salah satu modul, persegi atau persegi panjang. Dan dapat di cetak sebagai kode inversi hitam putih atau kode pola *dot* (bagian penanda). Versi terbesar dari kode ini dapat mencapai 61 (422x 422 modul), yang dapat menyimpan 40.000 angka (qr*code*.com, 2013).

d) SQRC

Jenis *QR Code* ini dilengkapi dengan membaca fungsi pembatas. Ini dapat digunakan untuk menyimpan informasi pribadi untuk mengelola informasi internal perusahaan dan sejenisnya (qr*code*.com, 2013).

e) LogoQ

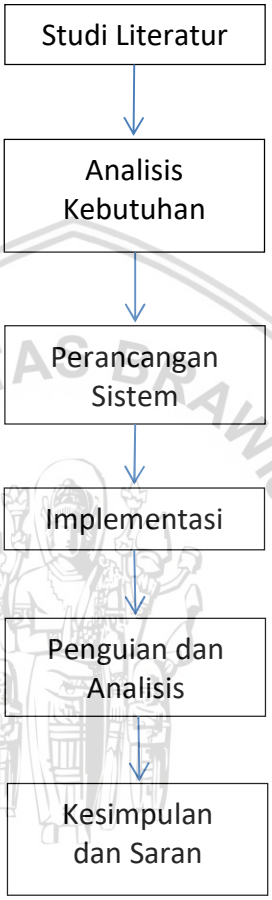
Gambar 2.19 Contoh *LogoQ*

Sumber : (qrcode.com, 2013)

Jenis *QR Code* yang dapat menggabungkan fitur desain tingkat tinggi seperti ilustrasi, huruf dan logo. *QR Code* ini menggunakan Logika *Since proprietary* (qrcode.com, 2013).

BAB 3 METODOLOGI

Bab ini menjelaskan metode yang digunakan dalam penelitian, yaitu studi literatur, fungsi algoritme SPECK, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis, dan pengambilan kesimulan dan saran. Runtutan pengerjaan penelitian dapat dilihat pada diagram alir berikut:



Gambar 3.1 Diagram alir penelitian

Jenis penelitian yang digunakan pada penelitian ini adalah penelitian pengembangan. Menurut Sugiyono (2014) penelitian pengembangan merupakan metode penelitian yang digunakan untuk menghasilkan produk tertentu dan menguji keefektifan suatu produk .Dapat disimpulkan, penelitian pengembangan diawali dengan penelitian skala kecil yang bisa dalam bentuk pengumpulan data terhadap permasalahan yang dihadapi dan akan dicari solusinya.

3.1 Studi Literatur

Metode ini digunakan untuk mendapatkan dasar teori sebagai sumber acuan untuk penulisan skripsi dan pengembangan aplikasi. Teori dan pustaka yang berkaitan dengan tugas akhir ini meliputi:

1. Proses enkripsi dan dekripsi
2. Algoritme SPECK 64 bit
3. PHP.
4. QR-*Code*

3.2 Analisis Kebutuhan

Analisis kebutuhan perangkat lunak meliputi analisis spesifikasi perangkat lunak. Metode analisis menggunakan bahasa pemodelan UML (*Unified Modeling Language*). *Use Case Diagram* digunakan untuk mendeskripsikan kebutuhan-kebutuhan dan fungsionalitas sistem dari perspektif *user*. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan sistem yang akan dimodelkan dalam diagram *use case*. Kebutuhan fungsional yang nantinya akan disediakan oleh aplikasi yang dibangun antara lain adalah:

1. Sistem harus mengidentifikasi QR *code*.
2. Sistem dapat menyediakan QR *Code*.
3. Sistem dapat mengimplementasikan enkripsi dan dekripsi pada QR *Code*.
4. Sistem harus mampu menerima inputan semua jenis karakter berupa *string*, huruf, angka dan simbol
5. Sistem harus mampu merubah karakter menjadi QR *Code*.
6. Sistem harus mampu mengimplementasikan algoritme SPECK untuk enkripsi dan dekripsi pada QR- *Code*.

3.3 Perancangan Sistem

Perancangan arsitektur sistem yang digunakan dalam penelitian ini dilakukan setelah semua kebutuhan untuk pembuatan sebuah web yang didapatkan melalui tahap analisis kebutuhan telah terpenuhi. Perancangan dimulai dari perancangan umum sistem keamanan webi dengan memodelkan gambaran terhadap lingkungan aplikasi, perancangan basis data dengan cara identifikasi *class*, perancangan alur aktifitas yang dimodelkan dalam *activity diagram* dan selanjutnya alur perancangan terhadap sisi keamanan web sebagai fokus penelitian.

3.4 Implementasi

Implementasi dilakukan dengan mengacu kepada perancangan perangkat lunak. Implementasi diawali dengan penjabaran lingkungan perangkat lunak yang akan digunakan. Pembuatan aplikasi dikembangkan secara spesifik pada sistem operasi windows yang dibuat.

3.5 Pengujian dan Analisis

Pengujian perangkat lunak dilakukan agar dapat diketahui apakah perangkat lunak dapat berjalan optimal sesuai dengan spesifikasi, tujuan, dan kebutuhan yang telah dirancang sebelumnya. Strategi pengujian perangkat lunak yang akan digunakan antara lain:

1. Pengujian *test vector* , untuk mengetahui apakah hasil *ciphertext*, *plaintext* dari program sesuai dengan hasil dari journal "*THE SIMON AND SPECK FAMILIES OF LIGTHWEIGHT BLOCK CIPHER*", *plaintext* akan diubah menjadi *ciphertext* sesuai *test vector*, algoritme SPECK akan menjadi upaya sebagai pencegahan terhadap adanya modifikasi atau pengubahan dari pihak luar yang berusaha mengubah hasil sebuah kode.
2. Pengujian Fungsional, yang merupakan teknik pengujian yang berfokus pada keluaran hasil dari *respon* masukan, atau secara sederhana pengujian fungsional atau *black box* merupakan proses menjalankan aplikasi untuk mengetahui apakah ada error atau ada fungsi yang tidak berjalan sesuai harapan. black box ini mengabaikan mekanisme internal sistem, seperti bagaimana sistem bekerja memproses masukan.
3. Pengujian *Non*-Fungsioanal untuk mengetahui karakteristik kualitas komponen atau sistem diuji. *Non*-fungsional mengacu pada aspek perangkat lunak yang mungkin tidak berhubungan dengan fungsi atau petunjuk tindakan tertentu seperti skalabilitas atau keamanan.
4. Pengujian Keamanan untuk memastikan privasi kode dalam QR *Code* terjaga. Dengan metode SPECK maka sistem keamanan aplikasi akan memberi perlindungan fisik pada kode sehingga kode dapat diubah menjadi bentuk *ciphertext* yang sulit dimengerti isinya. Dengan metode ini juga akan meminimalkan celah bagi pihak lain untuk mengetahui isi informasi kode yang bukan termasuk kewenangannya, dengan didukung salah satu kamera dari perangkat keras yaitu *handphone* iOS versi 11 kamera akan mengenali *barcode* atau QR *code* secara otomatis, dan mengirimkan notifikasi ketika situs yang diidentifikasi benar. Berikut hasil pengujian kemanan.
5. Pengujian waktu enkripsi dan dekripsi dilakukan untuk mengetahui waktu eksekusi sistem pada proses enkripsi dan dekripsi, dengan menggunakan fungsi *microtime* agar mengetahui waktu tiap *rounds* dan jumlah waktu keseluruhan menggunaka *key* dan *plaintext* sesuai *test vector*.

Pada akhir pengujian akan dilakukan analisis pembahasan terhadap hasil ujian yang telah dilakukan. Analisis ini diperlukan untuk mengetahui keberhasilan aspek kerahasiaan, *confidentiality* yang sesuai dengan pengujian yaitu *Test Vector,* pengujian Fungsional, pengujian *Non*-Fungsional, Keamanan dari enkripsi dan dekripsi pada QR *code* dan pengujian waktu enkripsi dan dekripsi.

3.6 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dan saran dilakukan setelah semua tahapan metodologi penelitian telah selesai dilakukan. Kesimpulan diambil dari hasil uji dari implementasi algoritme speck untuk enkripsi dan dekripsi pada QR *Code*. Kesimpulan hasil uji dan analisis diambil bedasarkan parameter yang digunakan yang meliputi aspek kerahasiaan, *confidentiality* yang sesuai dengan pengujian yaitu *Test Vector,* pengujian Fungsional, pengujian *Non*-Fungsional, Keamanan dari enkripsi dan dekripsi pada QR *code* dan pengujian waktu enkripsi dan dekripsi. Pengujian dan analisis dari aplikasi dan dijabarkan apakah sistem dapat menyelesaikan permasalahan-permasalahan yang telah dirumuskan.

Penulisan saran bertujuan agar kekurangan-kekurangan dari sistem dapat diperbaiki dan memberikan pertimbangan untuk penelitian selanjutya dengan menggunakan metode atau algoritme yang berbeda agar dapat dibandingkan satu sama lain mana yang lebih baik.

BAB 4 PERANCANGAN

Bab ini membahas mengenai analisis dan perancangan sistem yang digunakan dalam penelitian implementasi algoritme*SPECK* untuk enkripsi dan dekripsi pada QC *code* berbasis web.

4.1 Algoritme SPECK

Simon dan Speck diusulkan secara terbuka pada bulan Juni 2013 oleh sekelompok peneliti di Direktorat Riset Keamanan Nasional AS. Karya desain dimulai pada tahun 2011, dan jumlah kriptanalisis yang signifikan telah dilakukan (tidak hanya oleh para *desainer*, namun oleh banyak orang di seluruh perusahaan) pada saat menjelang publikasi. Banyak cipher blok ringan telah diusulkan, dan banyak tampil dengan baik di berbagai *platform* terbatas. Simon dan Speck masing-masing memiliki beberapa instantiasi, yang mendukung ukuran blok 32, 48, 64, 96, dan 128 *bit*, dan dengan tiga ukuran kunci untuk disesuaikan dengan ukuran masing-masing blok. Setiap keluarga menyediakan sepuluh algoritme dalam semua. Gambar 2.1 mencantumkan blok dan ukuran kunci yang berbeda, dalam *bit*.

Tabel 4.1 Simon & Speck

Block size	*Key sizes*
32	*64*
48	*72,96*
64	*96,128*
96	*96,144*
128	*128,192,256*

Sumber : (Beaulieu, 2013)

Kesederhanaan desain memiliki keuntungan tambahan, Speck memiliki throughput tertinggi pada prosesor 64 bit dari setiap cipher blok yang diimplementasikan dalam perangkat lunak. Sebagian besar algoritme kriptografi yang dirancang untuk memenuhi kebutuhan komputasi dekstop pada saat ini. Bidang kriptografi yang lebih sederhana membahas keamanan dari suatu perangkat dengan sangat terbatas. Tujuan dari algoritme jenis ini adalah memberikan fleksibilitas dan karakteristik kinerja yang dibutuhkan pengembang. Algoritme jenis ini menekankan pada *blockcipher* tidak menyediakannya keamanan dengan sendirinya. Aplikasi yang berbeda juga akan memberikan tingkat keamanan yang berbeda dengan syarat tertentu serta pengembangan protokol yang harus dikembangkan sendiri-sendiri demi mencapai tingkat keamanan yang diinginkan. *Blockcipher* memiliki beberapa kegunaan dalam kriptogtafi primitif dan setiap protokol ringan bisa didasarkan pada *cipher* dengan ukuran yang tepat. Terdapat varian dari algoritme Speck berdasarkan

panjang kunci dan ukuran blok pesannya seperti dapat dilihat pada Gambar 2.2 dan Tabel 1 Berikut notasi yang digunakan pada algoritme Speck (Beaulieu, 2013):

Tabel 4.2 Notasi Algoritme Speck

⊕	Operator bitwise *XOR*
n	Ukuran panjang word pada SPECK (*n* = 16,24,32,48, atau 64)
+	Operator penjumlahan modulo 2^n
-	Operator pengurangan modulo 2^n
<<j	Rotasi bit ke kiri sebanyak *j* bit
>>j	Rotasi bit ke kanan sebanyak *j* bit
R	Jumlah *round*
a→b	Memperbarui nilai *a* dengan nilai *b*

Sumber :(Firmanesa, 2016)

Tabel 4.3 Varian algoritme Speck dan parameter yang digunakan

Sumber : (Firmanesa, 2016)

Block size $2n$	Key size mn	Word size n	Key words m	Rot α	Rot β	Rounds r
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3	8	3	26
	128		4			27
96	96	48	2	8	3	28
	144		3			29
128	128	64	2	8	3	32
	192		3			33
	256		4			34

4.2 Manualisasi SPECK 64– 128 Bedasarkan Test Vector

Keterangan Jenis Speck :

TIPE = 64 – 128

Block size (2n) = 64, Key Size (mn) = 128

T (PUTARAN) = 27, ALPHA = 8, BETA = 3, m (key word) = 4, n (word size)= 32

Data Test Vector :

Speck64/128

Key: 1b1a1918 13121110 0b0a0908 03020100

Plaintext: 3b726574 7475432d

Ciphertext: 8c6fa548 454e028b

CATATAN :

1. Penggunaan nilai 24 pada pergeseran kiri didapat dari:
n – alpha = 32 bit – 8 bit = 24 bit (nilai ini digunakan jika nilai pergeseran menghasilkan bilangan UNSIGN BIT).
2. Penggunaan nilai 29 pada pergeseran kanan didapat dari:
n – beta = 32 bit – 3 bit = 29bit (nilai ini digunakan jika nilai pergeseran menghasilkan bilangan UNSIGN BIT).

PROSES PERHITUNGAN :

PLAIN TEXT : 3b726574, 7475432d

KEY: 1b1a1918, 13121110, 0b0a0908, 03020100

*****PROSES KEY EKSPANSI*****

KEY EXPANDED KE 0 = KEY [3] = 03020100 =
11000000100000000100000000

KEY EXPANDED KE 0: 11000000100000000100000000

key[2] = ((key[2] << 24 | key[2] >>> 8)) + k) XOR i(0)
key[2] = ((10110000101000001001000001000 << 24 |
10110000101000001001000001000 >>> 8)) +
11000000100000000100000000) ^ 0
k = (k << 3 | (k >>> 29))^ key[2]
k = (11000000100000000100000000 << 3 |
(11000000100000000100000000 >>> 29))^
10110000110100001011000001001
KEY EXPANDED KE 0: 10011000111010000001100001001

key[1] = ((key[1] << 24 | key[1] >>> 8)) + k) XOR i(1)
key[1] = ((10011000100100001000100010000 << 24 |
10011000100100001000100010000 >>> 8)) +
10011000111010000001100001001) ^ 1
k = (k << 3 | (k >>> 29))^ key[1]
k = (10011000111010000001100001001 << 3 |
(10011000111010000001100001001 >>> 29))^
10001100110000000101010011011
KEY EXPANDED KE 1: 10111011110110000000110101010011

Tabel 4.4 Hasil manualisasi ekspansi

KEY EXPANDED KE 2	11010011001101001101111110011
KEY EXPANDED KE 3	11111111010010000110101011100101
KEY EXPANDED KE 4	11001111110011011001110010101
KEY EXPANDED KE 5	11101001100011001011001111010010
KEY EXPANDED KE 6	10101010110001110110110010111101
KEY EXPANDED KE 7	11111110101100101010000111001000
KEY EXPANDED KE 8	1111111010100000101100010
KEY EXPANDED KE 9	11000100110101001100111010110 1
KEY EXPANDED KE 10	11011111111101110000100010000010
KEY EXPANDED KE 11	10011110010010000111110010010011
KEY EXPANDED KE 12	10101001001101001011100100101000
KEY EXPANDED KE 13	11011101001011101101111011110101
KEY EXPANDED KE 14	10001011111001100011100010001101
KEY EXPANDED KE 15	11111011100000110101110001001
KEY EXPANDED KE 16	10101110000111101010101111111000
KEY EXPANDED KE 17	10010110101110110110000010111
KEY EXPANDED KE 18	11011101010110011001101011011 00
KEY EXPANDED KE 19	11010100001101010111001000 10010
KEY EXPANDED KE 20	10000101111000110101111001010
KEY EXPANDED KE 21	11000000101011111011101001 10010
KEY EXPANDED KE 22	11010011110010011011001110000001
KEY EXPANDED KE 23	10110011010001111000000100111101
KEY EXPANDED KE 24	10001100000100010011110000110101
KEY EXPANDED KE 25	11111110011010110101001000111010
KEY EXPANDED KE 26	11000010110000101100110110011010

*****PROSES ENKRIPSI*****

NILAI AWAL X = 3b726574 = 11101101110010011001010111010 0, Y = 7475432d = 11101000111010101000011001011 01

x = ((x << 24 | (x >>> 8))+ y)^ key[i]
x = ((11101101110010011001010111010 0<< 24 | (11101101110010011001010111010 0 >>> 8))+ 11101000111010101000011001011 01)^ 11000000100000000100000000
y = (y << 3 |(y >>> 29)) ^ x
y = (11101000111010101000011001011 01 << 3 |(11101000111010101000011001011 01 >>> 29)) ^11101011101100101011010010010010

NILAI X, Y PADA ROUND KE 1:
11101011101100101011010010010010,10010000001100010101101111 11001

x = ((x << 24 | (x >>> 8))+ y)^ key[i]

```
x = ( (11101011101100101011010010010010<< 24 |
(11101011101100101011010010010010 >>> 8) )+
10010000001100010101101111111001)^
10011000111010000001100001001
y = (y << 3 |(y >>> 29) ) ^ x
y = (10010000001100010101101111111001 << 3
|(10010000001100010101101111111001 >>> 29) )
^11001000000110010110001110100100
```

Tabel 4.5 Hasil manualisasi enkripsi

ROUND	NILAI X	NILAI Y
2	11001000000110010110001110100100	10001000110111000000110001101110
3	10010110011111000010100010000010	11010000100111000100101111110110
4	10111100000000110000101111101101	11011010111000111101101001011011
5	10110111111001011110111010000101	11000001111101100111100010110011
6	10000001010101011110110000011100	10000110100011000000111011000111
7	10010101000000111010111011000001	11111101110000110100001010111101
8	10010101110101101001111100010001	10111101111001100100010101100010
9	11100001101000011000010111100 1	11010111010110100100101001101100
10	10100110110100000110000000001111	11101001101110100110001101101001
11	11001000001110001111100000110100	10000101111010111110001101111011
12	11001010100001100010100111100 01	10010100001110000000111100101101
13	10100101110010010010111011010010	11110101001010010101011110111000
14	11011101111101110011001110011110	11000111101100010010010000001001
15	10010110000111011000001010101 11	11101101000011111110000100011001
16	10001100011010011010111010101 11	11110010000010111101111110011100
17	10101100010001001111111111111010	11000110011111001000001100011101
18	11101011010101010000111101100100	11011000101100010001011110001010
19	10111101001011000000010001110	11101010110000111011110011011000
20	10110010111111100101010110100	10000000100001000101100011100 11
21	10011110010000100011010100101111	10011100010100110101011010110101
22	11011011010011011111001100100000	11100111010111010000110100011100
23	11011011010011011111001100 10000	11100111010111010000110100011100
24	11101011100101010010010100 1101	11110100010111110111110100101100
25	10010110011111010000110101011001	10011001111001100100111000000
26	11100000110000100111101011111000	11110010010010000110100111111000
27	10001100011011111010010101001000	10001010100111000000010100010 11
Hasil Akhir (Biner)	10001100011011111010010101001000	10001010100111000000010100010 11
Hasil Akhir (Hexa)	8c6fa548	454e028b

*****PROSES DEKRIPSI*****

NILAI AWAL X = 10001100011011111010010101001000, Y =
10001010100111000000010100010 11

NILAI AWAL X HEXA= 2d31393338383430323438, Y =
31313632373430333633

PROSES ROUND KE 27

y = x ^ y
y = 10001100011011111010010101001000 ^
10001010100111000000010100010 11
y = 11001001001000011010011111000011
y = y << 29 | (y >>> 3)
y = 11001001001000011010011111000011 << 29 |
(11001001001000011010011111000011 >>> 3)
y = 11110010010010000110100111110 00
x = (x ^ key[26]) - y
x = (10001100011011111010010101001000 ^
11111110011010110101001000111010) -
11110010010010000110100111110 00
x = 11111000111000001100001001111010
x = x << 8 | (x >>> 24)
x = 11111000111000001100001001111010 << 8 |
(11111000111000001100001001111010 >>> 24)
x = 11100000110000100111101011111000

NILAI X, Y PADA ROUND KE 27:
11100000110000100111101011111000,11110010010010000110100111
11000

PROSES ROUND KE 26

y = x ^ y
y = 11100000110000100111101011111000 ^
11110010010010000110100111110 00
y = 10011001111001100100111000000000
y = y << 29 | (y >>> 3)
y = 10011001111001100100111000000000 << 29 |
(10011001111001100100111000000000 >>> 3)
y = 10011001111001100100111000000
x = (x ^ key[25]) - y
x = (11100000110000100111101011111000 ^

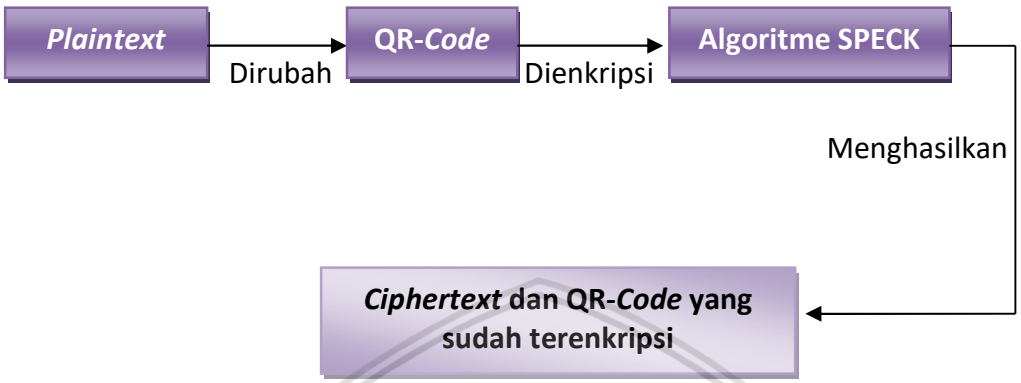
```
10001100000100010011110000110101) -
10011001111001100100111000000
x = 10110011001011001111110100001101
x = x << 8 | (x >>> 24)
x = 10110011001011001111110100001101 << 8 |
(10110011001011001111110100001101 >>> 24)
x = 10010110011111101000011010101l001

NILAI X, Y PADA ROUND KE 26:
10010110011111101000011010101l001,10011001111001100100111000000
```

Tabel 4.6 Hasil manualisasi dekripsi

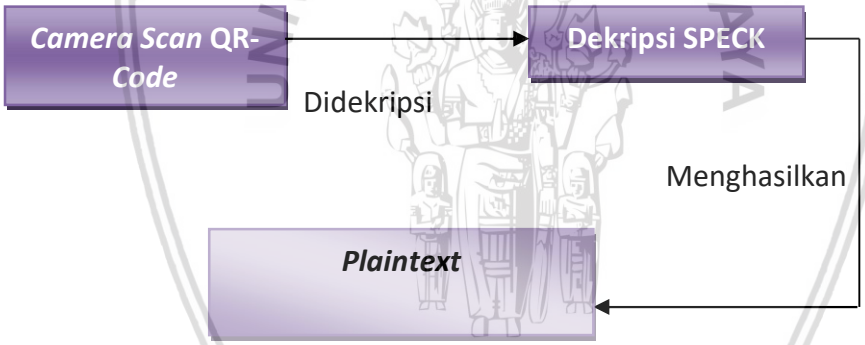
ROUND	X	Y
25	10010010010100111101000111110100	11000010101000001110001001O011
24	11101011100101010010010100l101	11110100010111110111110100101100
23	11011011010011011111001100100000	11100111010111010001101000l100
22	10011110010000100011010100101111	10011100010100110101011010110101
21	10110010111111100101010110100	10000000100001000101100011100l1
20	10111101001011000000010001110	11101010110000111011110011011000
19	11101011010101010000111101100100	11011000101100010001011110001010
18	10101100010001001111111111111010	11000110011111100100000110001l101
17	10001100011010011010111010101l1	11110010000010111101111110011100
16	10010110000111011000001010101l1	11101101000011111100001000110011
15	11011101111101110011001110011110	11000111101100010010010000001001
14	10100101110010010010111011010010	11110101001010010101011110111000
13	11001010100001100010100111100l1	10010100001110000001111100101101
12	11001000001110001111100000110100	10000101111010111110001101111011
11	10100110110100000110000000011111	11101001101110100110001101101001
10	11100001101000011000010111100l	11010111010110100100101001101100
9	10010101110101101001111100010010	10111101111001100100010101100010
8	10010101000000111010111011000l1	11111101110000110100001010111101
7	10000001010101011110110000011100	10000110100011000000111011000111
6	10110111111001011110111010000101	11000001111101100111100010l1011
5	10111100000000110000101111011101	11011010111000111101101001011011
4	10010110011111000010100010000010	11010000100111000100101111110110
3	11001000000110010110001110100100	10001000110111000000110001101110
2	11101011101100101011010010010010	10010000001100010101101111111001
1	11101101110010011001010111010O	11101000111010101000011001011O1
Hasil Akhir (Biner)	11101101110010011001010111010O	11101000111010101000011001011O1
Hasil Akhir (Hexa)	3b726574	7475432d

4.3 Perancangan Umum Sistem

Perancangan umum sistem aplikasi merupakan tahap awal dari perancangan perangkat lunak. Perancangan dilakukan dengan merepresentasikan arsitektur sistem secara umum, seperti yang ditunjukkan pada Gambar 4.3 dan Gambar 4.4



Gambar 4.1 Perancangan Enkripsi



Gambar 4.2 Perancangan Dekripsi

Pada Gambar 4.3 merupakan enkripsi yang berfungsi untuk merubah dari kode asli menjadi QR *Code* setelah proses *chypertext*. Selanjutnya pada Gambar 4.4 berfungsi sebagai dekripsi yang bermula dari gambar QR *Code* di *scan* lalu dimasukkan ke dalam dekripsi SPECK, dekripsi ini berfungsi untuk membalikkan kode yang terenkripsi menjadi kode asli sehingga dapat terbaca.

4.4 Analisis Kebutuhan

Bagian analisis kebutuhan menjelaskankebutuhan fungsional, *non*-fungsional, spesifikasi dan manajemen kebutuhan. Analisis kebutuhan Fungsional sangat diperlukan agar penulis bisa memberikan gambaran aplikasi yang dibuat

serta menjadi dokumentasi yang akan membantu pengembangan aplikasi selanjutnya.

4.4.1 Kebutuhan Fungsional

1. Sistem harus mampu menerima inputan berupa kode.
2. Sistem harus mampu melakukan enkripsi dengan algoritme SPECK.
3. Sistem harus mampu mengubah kode menjadi Qr-*code*.
4. Sistem harus mampu melakukan Scan Qr-*code*.
5. Sistem harus mampu melakukan dekripsi dengan algoritme SPECK.

4.4.2 Kebutuhan *Non* Fungsional

1. Portability : Sistem harus dapat berjalan pada semua web browser moderen.

4.4.3 Spesifikasi dan Manajemen Kebutuhan

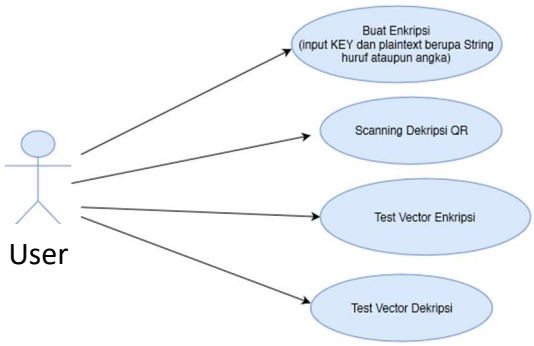
1. Memasukkan KEY (SED_1_001
2. Memasukan *plaintext* (SED_1_002)
3. Melakukan Enkripsi SPECK (SED_1_003)
4. Merubah Kode Menjadi Qr-*Code* (SED_1_004)
5. Melakukan Scan Qr-*Code* (SED_1_005)
6. Melakukan Dekripsi SPECK (SED_1_006)
7. Melakukan *Test Vector* Enkripsi (SED_1_007)
8. Melakukan *Test Vector* Dekripsi (SED_1_008)

4.5 Pemodelan Analisis Kebutuhan

Pemodelan kebutuhan menjelaskan *Use case* diagram, *use case* scenario, *activity* diagram, *sequence* diagram dan *class* diagram.

4.5.1 *Use case* diagram

Usecase Diagram merupakan salah satu model UML yang digunakan untuk mendeskripsikan kebutuhan fungsional sebuah sistem dari perspektif *end user*. *Use case* juga menunjukan perilaku (*behavior*) dari sistem yang akan dibuat. Diagram *use case* pada sistem ini dapat dilihat dalam Gambar 4.5.



Gambar 4.3 *Use case* Diagram Enkripsi, Dekripsi dan *Test Vector*

Diagram *use case* pada Gambar 4.5 merupakan proses enkripsi, dekripsi dan *test vector* yang mana user dapat memasukkan *key,*memasukkan *plaintext,* berupa *String* huruf ataupun angka,melakukan enkripsi SPECK dan merubah plaintext menjadi QR-Code, setelah itu pada tahap dekripsi yang mana user dapat memasukkan *key,* melakukan scan QR-Code, dan melakukan dekripsi SPECK, dan pada tahapan *test vector* enkripsi user dapan memasukkan key, plaintext sesuai aturan SPECK 64-128 begitupun pada *test vector* dekripsi user dapat melakukan *scan* QR-*code* dengan memasukkan *key* sesuai aturan *test vector* SPECK 64-128 terlebih dahulu.

4.5.2 *Use case* Skenario

Skenario setiap bagian pada *use case* menunjukkan proses apa yang terjadi pada setiap bagian didalam *use case* tersebut, dimana *Actor* memberikan perintah pada setiap bagian dan respon apa yang diberikan oleh sistem kepada *Actor* setelah aktor memberikan perintah pada setiap bagian – bagian *use case*. Berikut penjelasan *use case* skenario aplikasi.

Tabel 4.7 *Use case* Skenario Memasukan key

Use case Skenario Memasukkan *KEY* (SED_1_001)

Actor	*User*
Objective	User dapat memasukan inputan berupa *KEY*.
Pre-condition	User telah mengakses website SPECK.
Main flow	1. User memilih menu “Enkripsi”. 2. Sistem menampilkan halaman Enkripsi. 3. User mengimputkan *KEY* atau Kunci yang di inginkan.
Alternative flows	-
Post-condition	*KEY* atau Kunci telah diinputkan ke dalam sistem.

Tabel 4.8 *Use case* Skenario Memasukan *Plaintext*

Use case Skenario Memasukkan *Plaintext* (SED_1_002)

Actor	*User*
Objective	*User* dapat memasukan inputan *plaintext* berupa String huruf ataupun angka .
Pre-condition	*User* telah mengakses *website* SPECK.
Main flow	1. *User* memilih menu "Enkripsi". 2. Sistem menampilkan halaman Enkripsi. 3. User mengimputkan *Plaintext* atau Kode yang di inginkan.
Alternative flows	-
Post-condition	Plain atau Kode telah diinputkan ke dalam sistem.

Tabel 4.9 *Use case* Skenario Melakukan Enkripsi

Use case Skenario Melakukan Enkripsi (SED_1_003)

Actor	*User*
Objective	*User* dapat melakukan Enkripsi Kode
Pre-condition	*User* telah mengakses *website* SPECK.
Main flow	1. *User* memilih menu "Enkripsi". 2. Sistem menampilkan halaman Enkripsi. 3. User mengimputkan *Plaintext* atau Kode yang di inginkan. 4. User mengklik tombol "*Encrypt*". 5. Sistem menampilkan hasil proses enkripsi mulai dari *round* 1-27.
Alternative flows	-
Post-condition	*Plain* atau kode telah dienkripsi oleh sistem.

Tabel 4.10 Usecase Skenario Merubah Kode Menjadi Qr-*Code*

Usecase Skenario merubah kode menjadi Qr-*Code* (SPK_1_004)

Actor	User
Objective	User dapat merubah kode menjadi Qr-*Code*.
Pre-condition	User telah melakukan proses enkripsi kode yang di ingnkan.
Main flow	1. User mengklik tombol "QR *Code*". 2. Sistem akan menampilkan gambar Qr-*Code*.
Alternative flows	-
Post-condition	Sistem telah menampilkan gambar Qr-*Code* kepada *user*.

Tabel 4.11 *Use case* Skenario Melakukan *Scan* Qr-*Code*

Use case Skenario melakukan scan Qr-*Code* (SED_1_005)

Actor	*User*
Objective	*User* dapat melakukan Scan Qr-*Code*.
Pre-condition	*User* telah melakukan proses enkripsi kode yang di ingnkan.
Main flow	1. *User* memilih menu dekripsi. 2. Sistem menampilkan halaman dekripsi. 3. *User* mengklik tombol "Nyalakan" 4. Sistem akan mengaktifkan fungsi kamera pada perangkat yang digunakan. 5. User menunjukan gambar dari Qr-*Code* yang telah dienkripsi sebelumnya.
Alternative flows	-
Post-condition	Gambar Qr-*Code* telah terbaca oleh *system*.

Tabel 4.12 *Use case* Skenario melakukan Dekripsi

Use case Skenario melakukan Dekripsi (SED_1_006)

Actor	*User*
Objective	*User* dapat melakukan Dekripsi Kode
Pre-condition	*User* telah mengakses halaman Dekripsi.
Main flow	1. User memilih menu dekripsi. 2. Sistem menampilkan halaman dekripsi. 3. User mengklik tombol "Nyalakan" 4. Sistem akan mengaktifkan fungsi kamera pada perangkat yang digunakan. 5. User menunjukan gambar dari Qr-*Code* yang telah dienkripsi sebelumnya. 6. Sistemakan menampilkan proses dari dekripsi mulai dari *state* awal sampai akhir.
Alternative flows	-
Post-condition	*Plain* atau kode telah didekripsi oleh sistem.

Tabel 4.13 *Use case* Skenario melakukan *Test Vector* Enkripsi

Use case Skenario Melakukan *Test Vector* Enkripsi (SED_1_007)

Actor	*User*
Objective	*User* dapat melakukan Enkripsi Kode berupa bilangan *hexa* decimal sesuai *test vector* algoritme SPECK
Pre-condition	User telah mengakses *website* SPECK.

Main flow	1. User memilih menu “Enkripsi”. *2.* User mengimputkan *KEY* dan *Plaintext* atau kode berupa kode *hexa* sesuai *test vector* 3. User mengeklik tombol QR-Code dan menghasilkan *Ciphertext* 4. Sistem menampilkan hasil proses enkripsi mulai dari *round* 1-27.
Alternative flows	-
Post-condition	*Plain* atau kode telah dienkripsi oleh sistem berupa kode *hexa*.

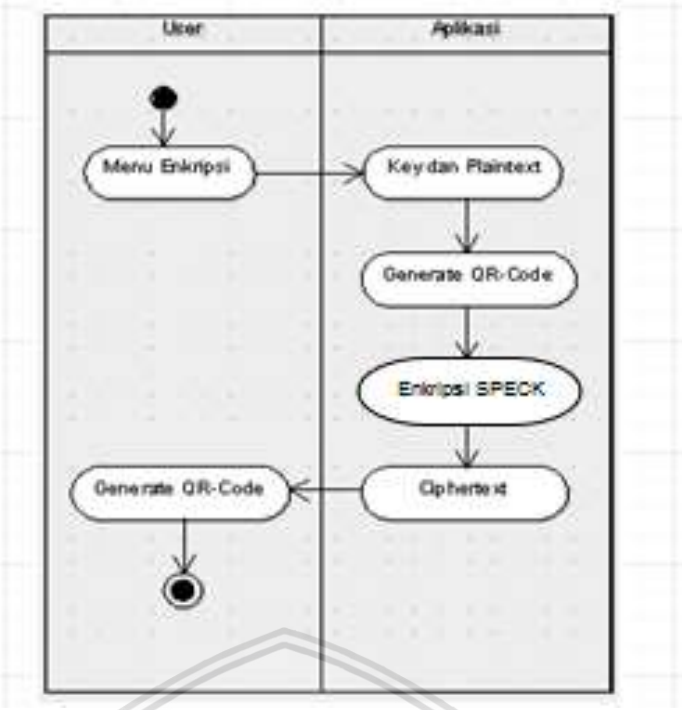
Tabel 4.14 *Use case* Skenario melakukan *Test Vector* Dekripsi

Use case Skenario Melakukan *Test Vector* Dekripsi (SED_1_008)

Actor	*User*
Objective	*User* dapat melakukan Dekripsi berupa QR-*Code* sesuai *test vector* algoritme SPECK dengan menginputkan KEY sesuai *test vector*
Pre-condition	User telah mengakses *website* SPECK.
Main flow	1. User memilih menu “Dekripsi”. 2. User memasukkan *KEY* sesuai test vector dan men-*scan* QR-*Code* 3. User mengeklik tombol QR-*Code* dan menghasilkan *Ciphertext* 4. Sistem menampilkan hasil proses dekripsi mulai dari *round* 1-27.
Alternative flows	-
Post-condition	*Plain* atau kode telah di dekripsi oleh sistem sesuai *test vector*

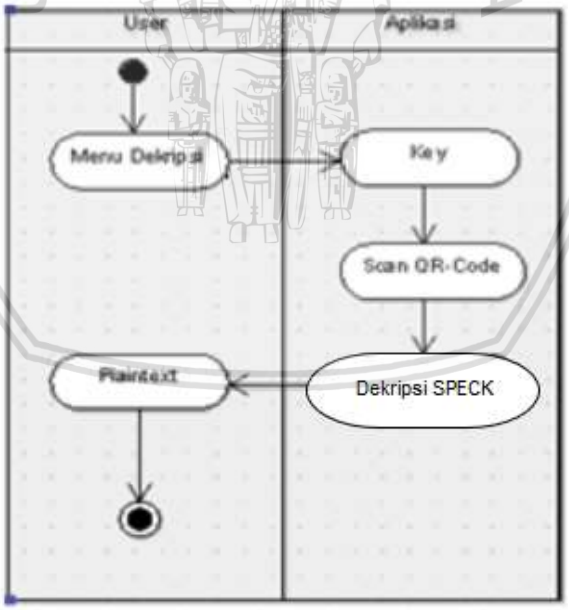
4.5.3 Activity Diagram

Activity diagram adalah tipe khusus dari diagram *state* yang memperlihatkan aliran dari suatu aktifitas ke aktifitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek. Sebuah aktifitas dapat direalisasikan oleh satu *use case* atau lebih. Aktifitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Berikut *activity diagram.*



Gambar 4.4 *Activity diagram* Enkripsi

Pada Gambar 4.6 *user* harus menekan tombol enkripsi lalu aplikasi meminta *key* dan *plaintext* untuk di rubah menjadi QR-*Code* kemudian dienkripsi sehingga menghasilkan *ciphertext* yang akan di *generate* kedalam QR-*Code*.



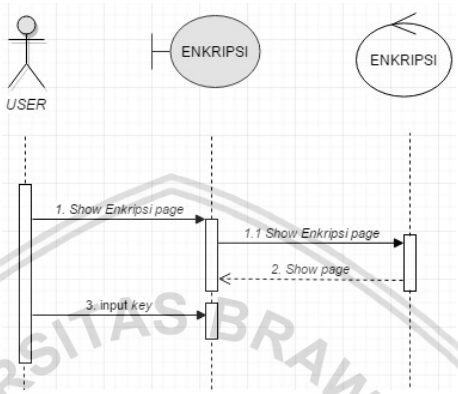
Gambar 4.5 *Activity diagram* Dekripsi

Pada Gambar 4.6, *user* harus menekan tombol dekripsi lalu aplikasi meminta scan QR-*code* untuk di dekripsi sehingga menghasilkan *plaintext*.

4.5.4 *Sequace* Diagram

Diagram *sequence* (urutan) adalah diagram interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu. Pada *Sequence diagram* ini menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan pada sistem sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu.

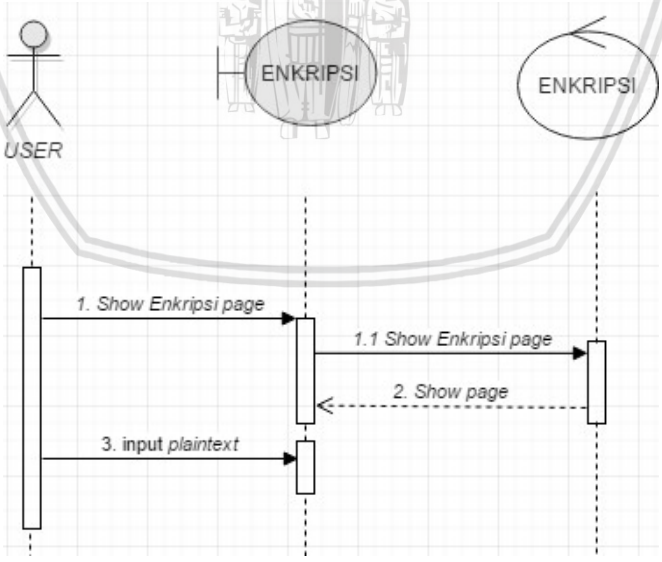
Sequence Diagram memasukkan *key* (SPK_1_001)



Gambar 4.8 *Sequence* Diagram memasukkan *key*

Pada gambar 4.8 merupakan *Sequence* diagram memassukan *key* atau langkah-langkah memasukkan *key* yang dilakukan pada sistem.

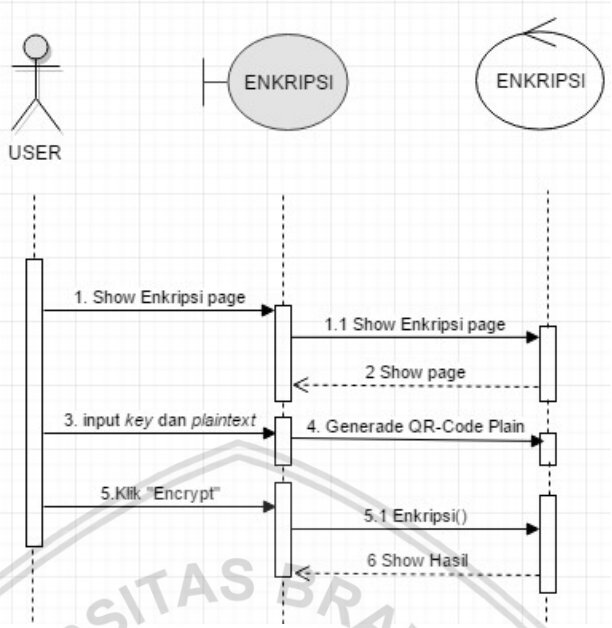
Sequence Diagram memasukkan *plaintext* (SPK_1_002)



Gambar 4.9 *Sequence* diagram memasukkan *plaintext*

pada gambar 4.9 merupakan *Sequence* diagram memasukan *plaintext* atau langkah-langkah memasukkan *plaintext* yang dilakukan pada sistem.

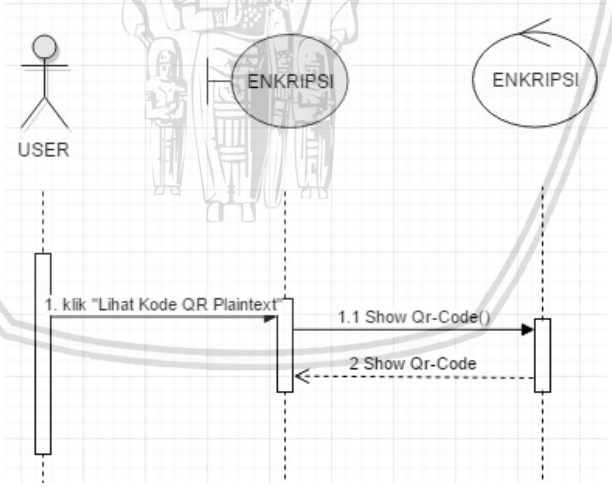
Sequence Diagram Melakukan Enkripsi (AED_1_003)



Gambar 4.10 *Sequence* Diagram Melakukan Enkripsi

Pada gambar 4.10 merupakan *Sequence* diagram melakukan enkripsi QR-*Code* atau langkah-langkah enkripsi QR-*Code* yang dilakukan pada sistem.

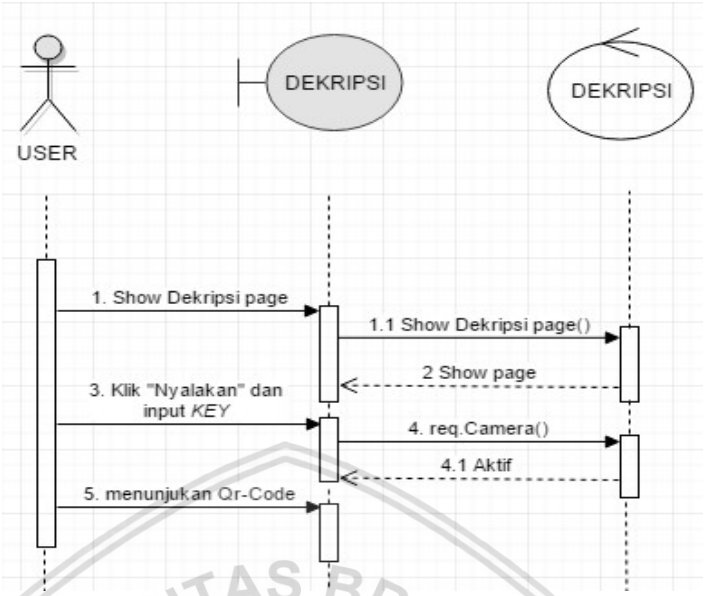
Sequence Diagram merubah *plaintext* menjadi QR-*Code* (SPK_1_004)



Gambar 4.11 *Sequence* Diagram merubah *plaintext* menjadi Qr-Code

pada gambar 4.11 merupakan *Sequence* diagram merubah *plaintext* menjadi QR-*Code* atau langkah-langkah merubah *plaintext* menjadi QR-*Code* yang dilakukan pada sistem.

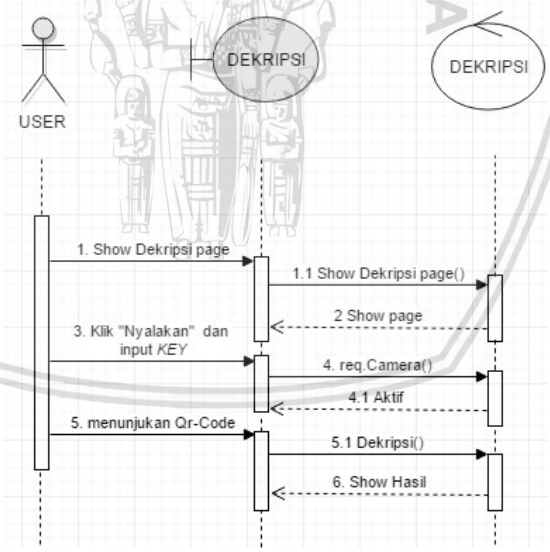
Sequence Diagram melakukan *scan* QR-*Code* (SPK_1_005)



Gambar 4.12 *Sequence* Diagram melakukan scan QR-Code

pada gambar 4.12 merupakan *Sequence* diagram melakukan scan QR-*Code* atau langkah-langkah melakukan scan QR-*Code* yang dilakukan pada sistem.

Sequence Diagram melakukan Dekripsi (SPK_1_006)

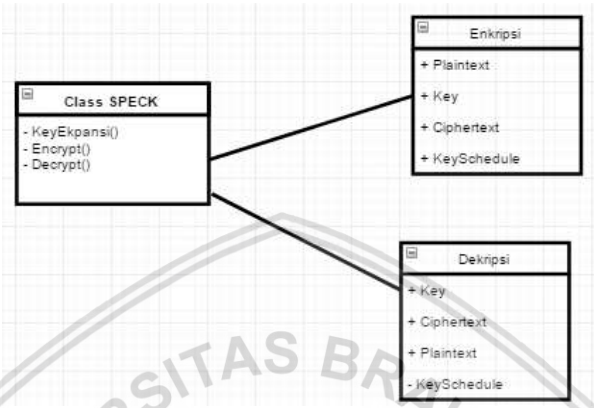


Gambar 4.13 *Sequence* Diagram melakukan Dekripsi

Pada gambar 4.13 merupakan Sequence diagram melakukan dekripsiQR-Codeatau langkah-langkah melakukan dekripsi QR-Codeyang dilakukan pada sistem.

4.5.5 *Class* Diagram

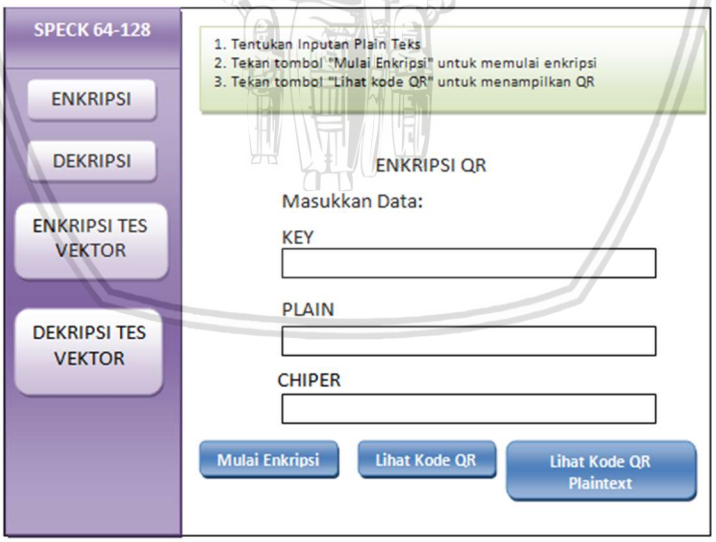
Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, dan relasi-relasi. Diagram ini umum ditemui pada pemodelan sistem berorientasi objek. Meski sifatnya statis, sering pula memuat kelas-kelas aktif. *Class diagram* aplikasi disajikan dalam Gambar 4.14



Gambar 4.14 Class Diagram

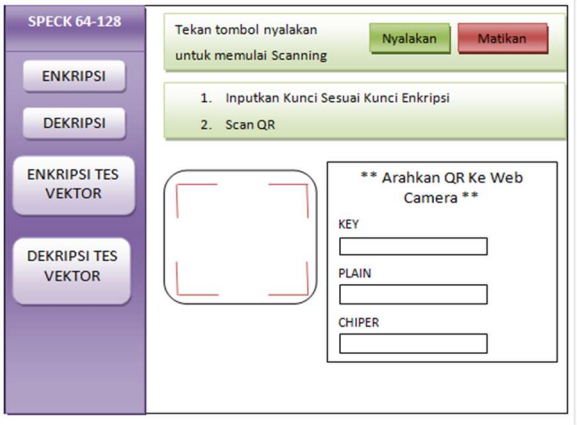
Pada gambar 4.14 merupakan gambar *class* diagram dari program SPECK yang mana memperlihatkan himpunan kelas-kelas,dan relasi-relasi.

4.6 Perancangan Antarmuka



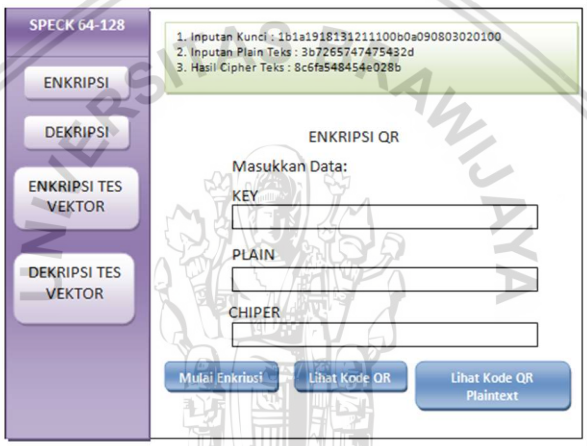
Gambar 4.6 Perancangan Antarmuka Enkripsi

Pada gambar 4.6 terdapat perancangan antar muka pada menu Enkripsi dengan memasukkan *key* sebanyak 16 karakter kemudian memasukkan *plaintext*.



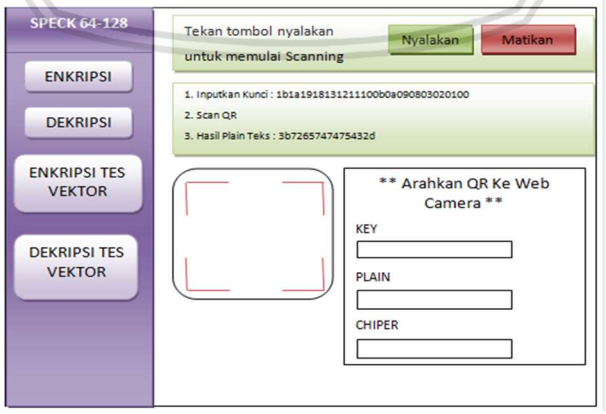
Gambar 4.7 Perancangan Antarmuka Dekripsi

Pada gambar 4.7 merupakan Perancangan Antarmuka pada menu Dekripsi dengan menyalakan kamera *scanning* untuk memulai proses dekripsi.



Gambar 4.8 Perancangan Antarmuka Enkripsi *Test Vector*

Pada gambar 4.8 merupakan perancangan antarmuka pada enkripsi *test vector* dengan memasukkan *key* dan *plaintext* sesuai aturan *test vecktor* SPECK 64-128.



Gambar 4.9 Perancangan Antarmuka Dekripsi *Test Vector*

Pada gambar 4.9 merupakan perancangan antarmuka pada dekripsi *test vector* dengan menyalakan kamera *scanner* dan memasukkan *key* sesuai aturan *test vector* pada SPECK 64-128.

4.7 Perancangan Pengujian

Tahapan pengujian akurasi dilaksanakan sebagaimana untuk mengukur tingkat keakuratan dari sistem yang mana pada tahapan pengujian ini menggunakan 5 metode pengujian yaitu pengujian *test vectors,* pengujian keamanan, pengujian *fungsional,* pengujiam *non fungsional, pengujian waktu enkripsi dan dekripsi.* untuk pengujian *test vectors* menggunakan sebuah *Key* dan *Plaintext* yang didapat dari *journal "THE SIMON AND SPECK FAMILIES OF LIGTHWEIGHT BLOCK"*, Hasil dari sistem akan di cocokkan dengan hasil dari manualisasi.

BAB 5 IMPLEMANTASI DAN PENGUJIAN

5.1 Implementasi Algoritme SPECK

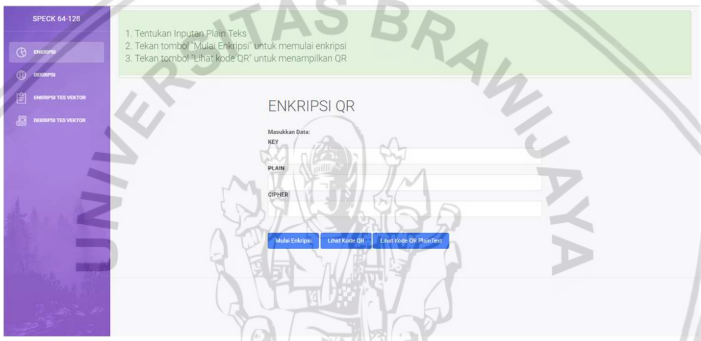
Kode sumber implementasi algoritme SPECK pada enkripsi dan dekripsi QR-*Code* dapat dilihat pada lampiran A Implementasi Algoritme SPECK.

5.2 Implementasi Antarmuka

Antarmuka digunakan *user* sebagai perantara interaksi terhadap sistem yang dijalankan untuk melakukan suatu aktifitas berupa perintah dari *user*. Pada sistem ini *user* dapat melakukan enkripsi dan dekripsi.

5.2.1 Implementasi Halaman Enkripsi

Halaman enkripsi merupakan halaman antarmuka bagi user untuk melakukan proses enkripsi pada sistem SPECK Enkripsi Dekripsi.



Gambar 5.1 Tampilan halaman enkripsi

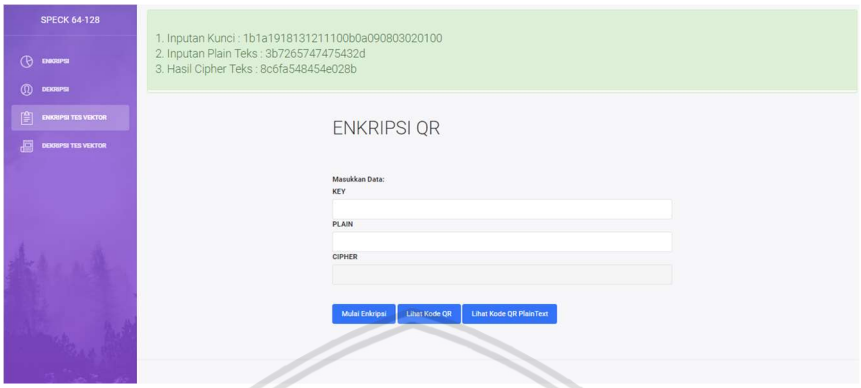
5.2.2 Implementasi Halaman Dekripsi

Halaman dekripsi merupakan halaman antarmuka bagi user untuk melakukan proses dekripsi pada sistem SPECK Enkripsi Dekripsi.

Gambar 5.2 Tampilan halaman dekripsi

5.2.3 Implementasi Halaman Enkripsi Pada *Test Vector*

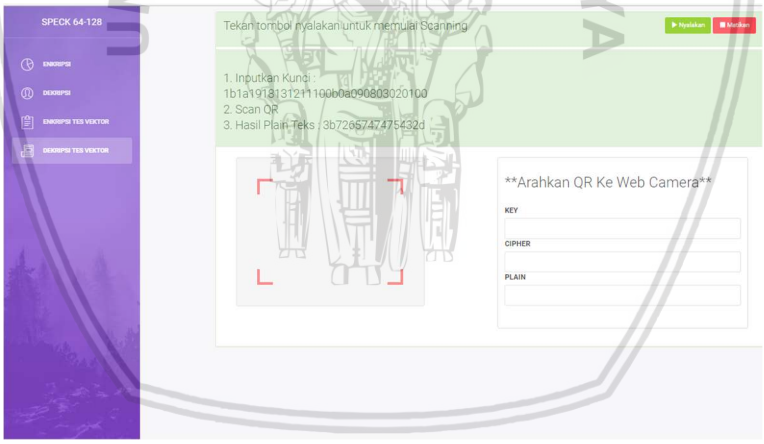
Halaman enkripsi pada *tes vector* merupakan halaman antarmuka bagi user untuk melakukan proses enkripsi pada *tes vector* sesuai aturan SPECK 64-128 pada sistem SPECK Enkripsi Dekripsi.



Gambar 5.3 Tampilan halaman enkripsi pada *test vector*

5.2.4 Implementasi Halaman Dekripsi Pada *Test Vector*

Halaman dekripsi pada *tes vector* merupakan halaman antarmuka bagi *user* untuk melakukan proses dekripsi sesuai *test vector* dengan memasukkan *key* sesuai dengan aturan SPECK 64-128 pada sistem SPECK Enkripsi Dekripsi



Gambar 5.4 Tampilan halaman dekripsi *test vector*

5.3 Pengujian

Poses pengujian terdiri dari 5 yaitu pengujian *test vectors,* pengujian fungsional dan pengujian *non*-fungsional, pengujian kemanan, pengujian waktu enkripsi dan dekripsi. Pada pengujian *test vectors* dilakukan dengan menginputkan sebuah *Key* dan *Plaintext* yang didapat dari *test vector* SPECK pada *journal "the simon and speck families of lighweight block cipher"* dan Hasil dari prosesnya akan di cocokkan dengan hasil dari manualisasi, setelah itu pengujian fungsional dan

pengujian *non*-fungsional dilakukan dengan metode *black box testing* dengan mencoba seluruh fungsional yang ada pada system, selanjutnya pengujian keamanan dilakukan dengan menggunakan salah satu kamera dari perangkat keras yaitu pada iOS 11 Iphone kamera akan mengenali *barcode* atau QR *code* secara otomatis, dan mengirimkan notifikasi ketika situs yang diidentifikasi benar, dan Pengujian waktu enkripsi dan dekripsi dilakukan untuk mengetahui waktu eksekusi sistem pada proses enkripsi dan dekripsi, dengan menggunakan fungsi *microtime* agar mengetahui waktu tiap *rounds* dan jumlah waktu keseluruhan menggunaka *key* dan *plaintext* sesuai *test vector*.

5.3.1 Pengujian *Test Vector*

Pada pengujian *test vectors* dilakukan dengan mengimputkan sebuah Key dan *Plaintext* yang didapat dari *test vector* SPECK sendiri dan Hasil dari prosesnya akan di cocokkan dengan hasil dari manualisasi.

Tabel 5.1 Pengujian *Test Vector*

SPECK *TEST VECTOR* BEDASARKAN *JOURNAL* "*THE SIMON AND SPECK FAMILIES OF LIGTHWEIGHT BLOCK CIPHER*"
Speck64/128 Key:1b1a191813121110 0b0a0908 03020100 Plaintext: 3b726574 7475432d Ciphertext: 8c6fa548 454e028b
PENGUJIAN *TEST VECTOR* DARI PROGRAM
ENKRIPSI **HASIL AKHIR X :**10001100011011111010010101001000 **HASIL AKHIR Y :**10001010100111000000010100001011 **HASIL AKHIR X (BENTUK *HEXA*):** 8c6fa548 **HASIL AKHIR Y (BENTUK *HEXA*):** 454e028b **HASIL AKHIR X BENTUK *STRING*:** Œo¥H **HASIL AKHIR Y BENTUK *STRING*:** EN‹ **HASIL AKHIR PENGGABUNGAN X DAN Y :** 8c6fa548454e028b (*Ciphertext*)
DEKRIPSI **HASIL AKHIR X :**11101101110010011001010111010 0 **HASIL AKHIR Y :**11101000111010101000011001011 01 **HASIL AKHIR X (BENTUK *HEXA*):** 3b726574 **HASIL AKHIR Y (BENTUK *HEXA*):** 7475432d **HASIL AKHIR PENGGABUNGAN X DAN Y :** 3b7265747475432d (*Plaintext*)

5.3.2 Pengujian Fungsional

Tabel 5.2 Pengujian fungsional

No	*Test Name*	*Test Cas*	*Expected Result*	Result	Status
1.	Pengujian Memasukka *KEY*	*user* mengimputkan sesuai aturan 16 karakter	Sistem merespon dengan menampilkan inputan halaman enkripsi/dekripsi.	Sistem merespon dengan menampilkan inputan halaman enkripsi/dekripsi.	*valid*
2.	Pengujian Memasukkan Kode/ plain	*user* mengimputkan kode	Sistem merespon dengan menampilkan inputan halaman enkripsi/dekripsi.	Sistem merespon dengan menampilkan inputan laman enkripsi/dekripsi.	*Valid*
3.	Pengujian Melakukan Enkripsi SPECK	*User* menekan tombol*"Encrypt"*	Sistem merespon dengan menampilkan proses enkripsi dari *round* 1-27.	Sistem merespon dengan menampilkan proses enkripsi dari *round* 1-27.	*valid*
4.	Pengujian Merubah Kode Menjadi QR-*Code* asli	*User* menekan tombol "QR-*Code*"	Sistem merespon dengan menampilkan QR-*Code*	Sistem merespon dengan menampilkan QR-*Code*	*Valid*
5.	Pengujian Merubah Kode Menjadi QR-*Code* yang sudah ter enkripsi	*User* menekan tombol "QR-*Code*"	Sistem merespon dengan menampilkan QR-*Code*	Sistem merespon dengan menampilkan QR-*Code*	*Valid*
6.	Pengujian Melakukan Scan QR-*Code*	*User* menekan tombol "nyalakan"	Sistem merspon dengan menampilkan hasil *Scan*.	Sistem merspon dengan menampilkan hasil *Scan.*	*Valid*
7.	Pengujian Melakukan Dekripsi SPECK	*User* memasukkan *KEY* sebanyak 16 karakter, lalu *User* menunjukan gambar QR-*Code* ke kamera perangkat yang sedang digunakan.	Sistem merespon dengan menampilkan proses dekripsi dari state awal sampai akhir.	Sistem merespon dengan menampilkan proses dekripsi dari state awal sampai akhir.	*Valid*
8.	Pengujian Melakukan Enkripsi *Test Vector* SPECK	*User* menekan tombol*"Encrypt"*dan mengetikkan inputan kunci, inputan *plaintext* sesuai *test vector* algoritme SPEECK	Sistem merespon dengan menampilkan proses enkripsi dari *round* 1-27.	Sistem merespon dengan menampilkan proses enkripsi dari *round* 1-27.	*Valid*
9.	Pengujian Melakukan Dekripsi *Test Vector* SPECK	*User* memasukkan *KEY sesuai tes vector* lalu *User* menunjukan gambar QR-*Code* ke kamera perangkat yang sedang digunakan.	merespon dengan menampilkan proses dekripsi dari state awal sampai akhir.	merespon dengan menampilkan proses dekripsi dari state awal sampai akhir.	*Valid*

5.3.3 Pengujian *Non*-Fungsional

Tabel 5.3 Pengujian *non*-fungsional

No	*Test Name*	*Test Cas*	*Expected Result*	Result	Status
1.	*Portability*	Sistem dijalankan di browser Google Chrome, Mozilla Firefox, dan Opera pada sistem operasi Windows.	SED berjalan tanpa masalah dan tampilan tidak berantakan	SED berjalan tanpa masalah dan tampilan tidak berantakan	*Valid*

5.3.4 Pengujian Keamanan

Pengujian keamanan dilakukan dengan menggunakan salah satu kamera dari perangkat keras yaitu *handphone* iOS versi 11 kamera akan mengenali *barcode* atau QR *code* secara otomatis, dan mengirimkan notifikasi ketika situs yang diidentifikasi benar. Berikut hasil pengujian kemanan.

Gambar 5.5 Qr-*Code Plaintext*

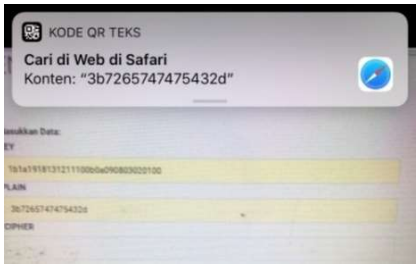
Pada gambar 5.5 merupakan gambar Qr-*code* yang belum terenkripsi oleh algoritme SPECK, pada aplikasi kamera terdapat konten "semangat untuk menggapai cita – cita!".

Gambar 5.6 Qr-*Code* sesudah di enkripsi

Pada Gambar 5.6 merupakan gambar QR-*Code* sesudah di enkripsi dengan algoritme SPECK, pada aplikasi kamera terdapat konten yang sudah terenkripsi.

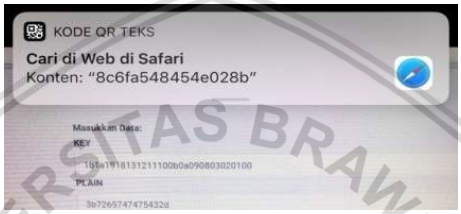
Gambar 5.7 Hasil Kode Qr-*Code* setelah di dekripsi

Pada gambar 5.7 merupakan bagian menu dekripsi untuk men-*scan* hasil QR-*Code* yang telah di enkripsi dengan memasukkan *key* terlebih dahulu sehingga *plaintext* asli dapat terbaca kembali.



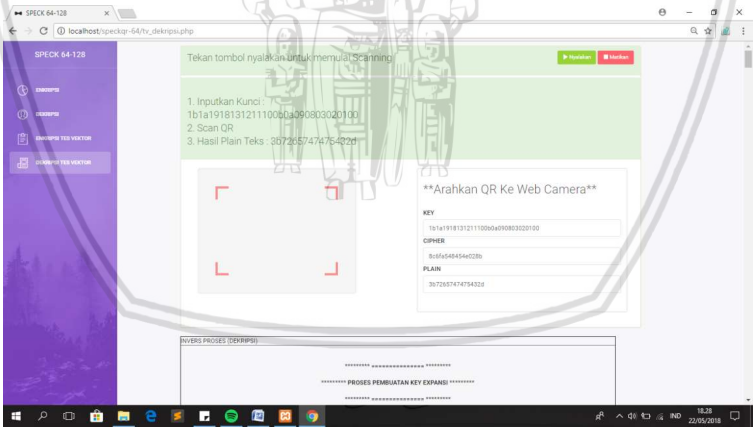
Gambar 5.8 Qr-*Code Plaintext Test Vector*

Pada gambar 5.8 merupakan hasil QR-*Code* pada *test vector* yang belum terenkripsi menggunakan algoritme sehingga pesan asli/*plaintext* masih dapat dibaca.



Gambar 5.9 *Test Vector* Qr-*Code* sesudah di enkripsi

Pada gambar 5.9 merupakan hasil QR-*Code* pada *test vector* yang sudah terenkripsi menggunakan algoritme SPECK, sehingga pesan asli tidak dapat terbaca.



Gambar 5.10 *Test Vector* Hasil Qr-*Code* setelah di dekripsi

Pada gambar 5.10 merupakan bagian menu dekripsi pada *test vector* untuk men-*scan* hasil QR-*Code* yang telah di enkripsi sehingga pesan asli/*plaintex*t dapat terbaca.

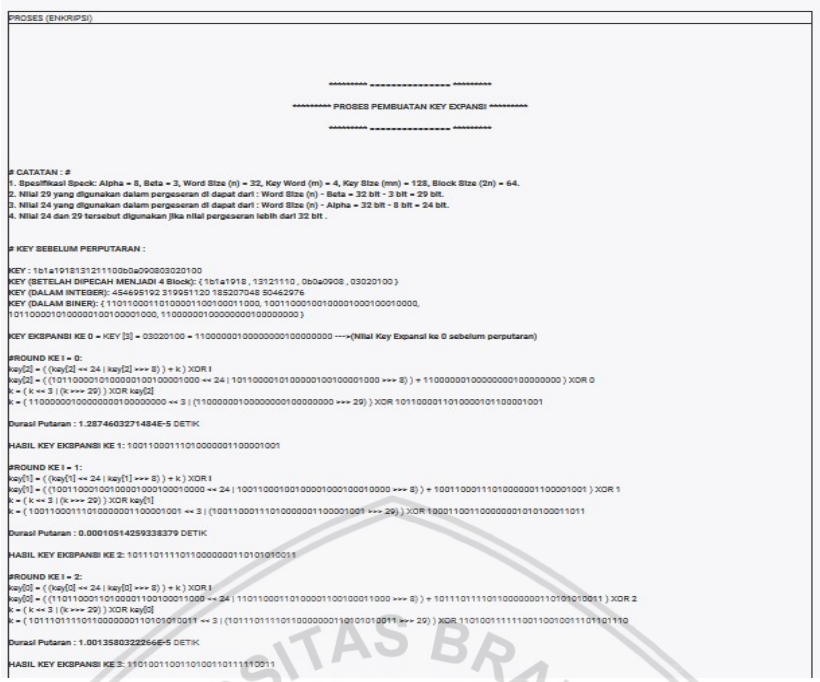
5.3.5 Pengujian Waktu Enkripsi dan Dekripsi

Pengujian waktu enkripsi dan dekripsi dilakukan untuk mengetahui waktu eksekusi sistem pada proses enkripsi dan dekripsi, dengan menggunakan fungsi *microtime* agar mengetahui waktu tiap *rounds* dan jumlah waktu keseluruhan menggunakan *key* dan *plaintext* berdasarkan *test vector*. Berikut kode sumber yang dibutuhkan untuk menampilkan hasil waktu tiap putaran, dimulai dari *round* 0 – 26 (27 putaran) pada SPECK ukuran 68/128 *bit.*

Kode sumber menampilkan waktu untuk tiap *rounds*

```
1   $start = microtime(TRUE);//Waktu mulai
2                       //--------------
3                       echo "<br><br> <b>#PROSES ROUND KE
4   ".($i+1)."</b>";
5                       echo "<br><br>y = x  XOR  y";
6                       echo "<br>y = ".decbin($x)." XOR ".decbin($y);
7                       $y = $x  ^  $y;
8                       echo "<br>y = ".decbin($y);
9                       echo "<br>y = y << 29 | (y >>> 3)";
10                      echo "<br>y = ".decbin($y)." << 29 |
11  (".decbin($y)." >>> 3)";
12                      $y = $y << 29 | $this->_rshift($y , 3);
13                      echo "<br>y = ".decbin($y);
14
15                      echo "<br>x = (x XOR key[$i]) - y";
16                      echo "<br>x = (".decbin($x)." XOR
17  ".decbin($key[$i]).") - ".decbin($y);
18                      $x = ($x ^ $key[$i]) - $y;
19                      echo "<br>x = ".decbin($x);
20
21                      echo "<br>x = x << 8 | (x >>> 24)";
22                      echo "<br>x = ".decbin($x)." << 8 |
23  (".decbin($x)." >>> 24)";
24                      $x = $x << 8 | $this->_rshift($x , 24);
25                      echo "<br>x = ".decbin($x);
26                      echo "<br><br> NILAI X, Y PADA ROUND KE
27  ".($i+1).": ". decbin($x). ",". decbin($y);
28                      //--------------
29                      $finish = microtime(TRUE); //Waktu selesai
30                      $totaltime = $finish - $start; //Durasi
31                      echo "<br><br><b>Durasi Putaran :
32  ".$totaltime."</b> DETIK<br>";
33                  }
34              $text[0] = $x; $text[1] = $y;
```

Berikut waktu yang dibutuhkan pada proses pembuatan *key* ekspansi pada proses enkripsi dengan menggunakan *key* dan *plaintext* sesuai *test vector* dari putaran ke 0 -25 yang ditunjukkan pada gambar 5.11 dan tabel 5.4



Gambar 5.11 Pengujian waktu pada *key* ekspansi proses enkripsi

Tabel 5.4 Pengujian waktu pada *key* ekspansi proses enkripsi

ROUND KE i = 3	7.8678131103516E-6 DETIK
ROUND KE i = 4	9.0599060058594E-6 DETIK
ROUND KE i = 5	8.1062316894531E-6 DETIK
ROUND KE i = 6	8.1062316894531E-6 DETIK
ROUND KE i = 7	7.8678131103516E-6 DETIK
ROUND KE i = 8	7.8678131103516E-6 DETIK
ROUND KE i = 9	8.1062316894531E-6 DETIK
ROUND KE i = 10	7.8678131103516E-6 DETIK
ROUND KE i = 11	8.1062316894531E-6 DETIK
ROUND KE i = 12	8.1062316894531E-6 DETIK
ROUND KE i = 13	8.1062316894531E-6 DETIK
ROUND KE i = 14	7.8678131103516E-6 DETIK
ROUND KE i = 15	7.8678131103516E-6 DETIK
ROUND KE i = 16	7.8678131103516E-6 DETIK
ROUND KE i = 17	3.6954879760742E-5 DETIK
ROUND KE i = 18	8.1062316894531E-6 DETIK
ROUND KE i = 19	9.0599060058594E-6 DETIK
ROUND KE i = 20	8.1062316894531E-6 DETIK
ROUND KE i = 21	8.1062316894531E-6 DETIK
ROUND KE i = 22	8.1062316894531E-6 DETIK
ROUND KE i = 23	9.0599060058594E-6 DETIK
ROUND KE i = 24	9.0599060058594E-6 DETIK
ROUND KE i = 25	9.0599060058594E-6 DETIK

Berikut waktu yang dibutuhkan pada proses enkripsi dengan menggunakan *key* dan *plaintext* sesuai *test vector* dari putaran ke 0 -26 yang ditunjukkan pada gambar 5.12 dan tabel 5.5

Gambar 5.12 Pengujian waktu proses enkripsi

Tabel 5.5 Pengujian waktu proses enkripsi

ROUND KE i = 3	7.8678131103516E-6 DETIK
ROUND KE i = 4	7.8678131103516E-6 DETIK
ROUND KE i = 5	3.4093856811523E-5 DETIK
ROUND KE i = 6	8.1062316894531E-6 DETIK
ROUND KE i = 7	9.0599060058594E-6 DETIK
ROUND KE i = 8	8.1062316894531E-6 DETIK
ROUND KE i = 9	8.1062316894531E-6 DETIK
ROUND KE i = 10	8.1062316894531E-6 DETIK
ROUND KE i = 11	8.1062316894531E-6 DETIK
ROUND KE i = 12	7.8678131103516E-6 DETIK
ROUND KE i = 13	7.8678131103516E-6 DETIK
ROUND KE i = 14	7.8678131103516E-6 DETIK
ROUND KE i = 15	7.8678131103516E-6 DETIK
ROUND KE i = 16	6.9141387939453E-6 DETIK
ROUND KE i = 17	8.1062316894531E-6 DETIK
ROUND KE i = 18	8.1062316894531E-6 DETIK
ROUND KE i = 19	3.0994415283203E-5 DETIK
ROUND KE i = 20	7.8678131103516E-6 DETIK
ROUND KE i = 21	8.1062316894531E-6 DETIK
ROUND KE i = 22	8.1062316894531E-6 DETIK
ROUND KE i = 23	7.1525573730469E-6 DETIK
ROUND KE i = 24	8.1062316894531E-6 DETIK
ROUND KE i = 25	8.1062316894531E-6 DETIK
ROUND KE i = 26	8.1062316894531E-6 DETIK
TOTAL WAKTU YANG DIBUTUHKAN	0.0007789134979248 DETIK

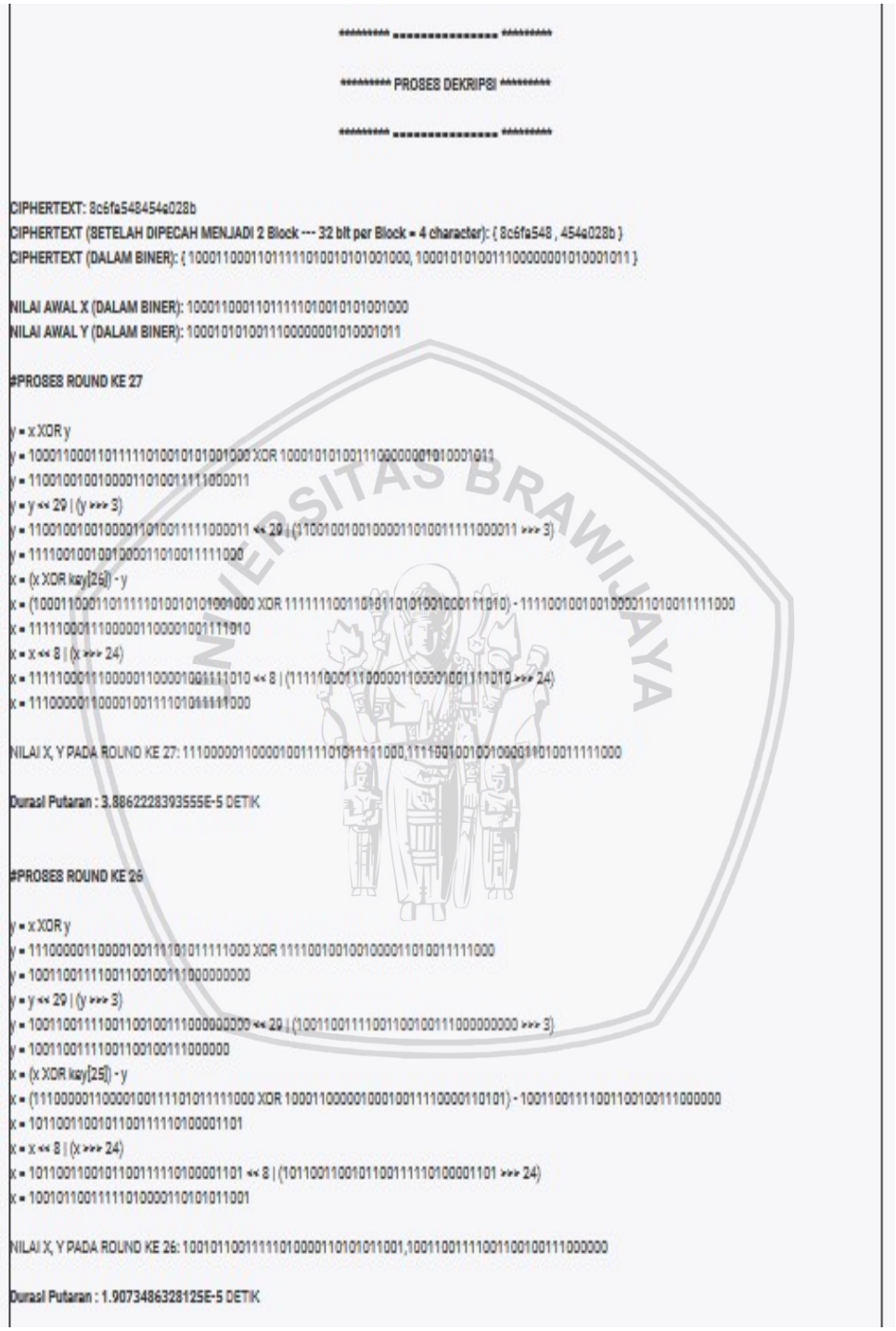
Berikut waktu yang dibutuhkan pada proses pembuatan *key* ekspansi pada proses dekripsi dengan menggunakan *key* dan plaintext sesuai *test vector* dari putaran ke 0 -25 yang ditunjukkan pada gambar 5.13 dan tabel 5.6

Gambar 5.13 Pengujian waktu pada *key* ekspansi proses dekripsi

Tabel 5.6 Pengujian waktu pada *key* ekspansi proses dekripsi

ROUND KE i = 3	8.8214874267578E-6 DETIK
ROUND KE i = 4	9.0599060058594E-6 DETIK
ROUND KE i = 5	9.0599060058594E-6 DETIK
ROUND KE i = 6	8.1062316894531E-6 DETIK
ROUND KE i = 7	7.8678131103516E-6 DETIK
ROUND KE i = 8	9.0599060058594E-6 DETIK
ROUND KE i = 9	8.1062316894531E-6 DETIK
ROUND KE i = 10	9.0599060058594E-6 DETIK
ROUND KE i = 11	1.9073486328125E-5 DETIK
ROUND KE i = 12	8.1062316894531E-6 DETIK
ROUND KE i = 13	7.1525573730469E-6 DETIK
ROUND KE i = 14	8.1062316894531E-6 DETIK
ROUND KE i = 15	7.8678131103516E-6 DETIK
ROUND KE i = 16	7.8678131103516E-6 DETIK
ROUND KE i = 17	7.8678131103516E-6 DETIK
ROUND KE i = 18	4.7922134399414E-5 DETIK
ROUND KE i = 19	9.0599060058594E-6 DETIK
ROUND KE i = 20	7.8678131103516E-6 DETIK
ROUND KE i = 21	9.0599060058594E-6 DETIK
ROUND KE i = 22	1.9073486328125E-5 DETIK
ROUND KE i = 23	7.8678131103516E-6 DETIK
ROUND KE i = 24	8.1062316894531E-6 DETIK
ROUND KE i = 25	8.1062316894531E-6 DETIK

Berikut waktu yang dibutuhkan pada proses pembuatan dekripsi dengan menggunakan key dan *plaintext* sesuai *test vector* dari putaran ke 27-1 yang ditunjukkan pada gambar 5.14 dan tabel 5.7



Gambar 5.14 Pengujian waktu proses dekripsi

Tabel 5.7 Pengujian waktu proses dekripsi

ROUND KE i = 25	1.5974044799805E-5 DETIK
ROUND KE i = 24	1.7881393432617E-5 DETIK
ROUND KE i = 23	0.00015091896057129 DETIK
ROUND KE i = 22	1.6927719116211E-5 DETIK
ROUND KE i = 21	1.6927719116211E-5 DETIK
ROUND KE i = 20	1.9073486328125E-5 DETIK
ROUND KE i = 19	1.8119812011719E-5 DETIK
ROUND KE i = 18	1.8119812011719E-5 DETIK
ROUND KE i = 17	1.7166137695312E-5 DETIK
ROUND KE i = 16	2.0027160644531E-5 DETIK
ROUND KE i = 15	1.6927719116211E-5 DETIK
ROUND KE i = 14	1.5974044799805E-5 DETIK
ROUND KE i = 13	7.0095062255859E-5 DETIK
ROUND KE i = 12	1.7166137695312E-5 DETIK
ROUND KE i = 11	1.5974044799805E-5 DETIK
ROUND KE i = 10	1.5974044799805E-5 DETIK
ROUND KE i = 9	1.7166137695312E-5 DETIK
ROUND KE i = 8	1.7881393432617E-5 DETIK
ROUND KE i = 7	1.6927719116211E-5 DETIK
ROUND KE i = 6	1.5020370483398E-5 DETIK
ROUND KE i = 5	1.6927719116211E-5 DETIK
ROUND KE i = 4	1.5974044799805E-5 DETIK
ROUND KE i = 3	6.103515625E-5 DETIK
ROUND KE i = 2	1.5974044799805E-5 DETIK
ROUND KE i = 1	1.5020370483398E-5 DETIK
TOTAL WAKTU YANG DIBUTUHKAN	0.01468300819397 DETIK

Pada pengujian waktu enkripsi dan dekripsi dapat disimpulkan bahwa total waktu yang dibutuhkan pada tiap *rounds* kurang dari 5 detik untuk tiap putarannya**.**

BAB 6 PENUTUP

Bagian ini memuat kesimpulan dan saran terhadap skripsi. Kesimpulan dan saran disajikan secara terpisah, dengan penjelasan sebagai berikut:

6.1 Kesimpulan

Bedasarkan hasil pengujian yang telah dilakukan dapat disimpulkan :

1. Cara SPECK mengimplementasi pada QR *Code* yaitu speck mengenkripsi plaintext dengan hasil berupa *ciphertext* yang dijadikan QR *Code.* QR *Code* sendiri dapat menampung data a-z dan 0 sampai 1 sehingga dapat memperbanyak *code* yang dapat dimasukkan sehingga tingkat keamanan lebih tinggi. Berdasarkan hasil pengujian, Kelebihan algoritme SPECK 64 *bit* dengan *key* 128 yang diimplementasikan pada sistem ini yaitu dapat berjalan baik di windows 32 *bit* dan 64 *bit* dan lebih efisien karena pada *prosessor* 64 *bit* memiliki *troughput* tertinggi dari setiap *blockcipher* yang diimplementasikan pada perangkat lunak, tujuan dari algoritme SPECK sendiri yaitu memberikan fleksibilitas dan karakterristik kinerja yang dibutuhkan oleh pengembang. Didukung dengan cara kerja algoritme SPECK sebagai berikut

   Pada proses pembuatan enkripsi:

   a. Memasukkan *Key* 128 *bit.*
   b. Memasukkan *plaintext.*
   c. Membuat *key* ekspansi sebanyak 27 Putaran.
   d. Membuat Enkripsi dengan melibatkan *key* ekspansi yang dihasilkan sebelumnya.
   e. Hasil Enkripsi berupa *ciphertext.*

   Pada proses pembuatan dekripsi:

   a. Masukan *Key* 128 *bit.*
   b. Scanning QR *Code* dari *ciphertext.*
   c. Membuat *Key* ekspansi sebanyak 27 putaran.
   d. Buat enkripsi dengan melibatkan *key* ekspansi yang dihasilkan sebelumnya.
   e. Hasil dekripsi berupa *plaintext.*
2. Pada proses validasi plaintext dan *ciphertext* pada algoritme SPECK dapat dibuktikan pada pengujian *test vector* yang membuktikan hasil *ciphertext* dan plaintext dari program maupun sistem sesuai dengan hasil dari *journal* "*THE SIMON AND SPECK FAMILIES OF LIGTHWEIGHT BLOCK CIPHER*" dengan melakukan pengecekan pada menu enkripsi *test vector* dan dekripsi *test vector* pada sistem yang menampilkan perhitungan untuk membuktikan bahwa pengujian *test vector* berhasil dilakukan.

3. Hasil kinerja pemrosesan enkripsi dan dekripsi algoritme SPECK pada Qr *Code* membuktikan bahwa algoritme SPECK adalah algoritme yang kuat dan cepat, dapat dibuktikan pada pengujian *daehader* untuk mengetahui variasi *key* yang di dapatkan mampu meningkatkan keamanan kerahasiaan informasi karena menghasilkan *key* yang unik sehingga menghindari terjadinya c*iphertext* yang sama dan waktu eksekusi yang diperlukan untuk melakukan enkripsi dan dekripsi pada tiap *rounds* kurang lebih 5 detik pada tiap putarannya yang didapatkan pada pengujian waktu enkripsi dan dekripsi.

6.2 .Saran

Berdasarkan hasil penelitian ditemukan beberapa permasalahan yang belum terpecahkan, sehingga peneliti mengajukan beberapa saran. Saran tersebut antara lain sebagai berikut:

1. Aplikasi ini dapat dikembangan lebih lanjut lanjut lagi, misalnya dengan memperbaiki tampilan yang lebih bagus, fitur yang lebih banyak.
2. Pada judul skripsi ini dapat dikembangkan menggunakan algoritme yang lain ataupun pengembangan berupa aplikasi pada sistem *android*, yang bertujuan untuk membandingkan mana yang lebih baik.

DAFTAR PUSTAKA

Ariadi, 2011. *Analisis Dan Perancangan Kode Matriks Dua Dimensi Quick Response (QR) Code.*. [online] Repository.usu.ac.id. Available at: <http://repository.usu.ac.id/handle/123456789/29816?show=full> [Accessed 30 June 2018].

Ariyus, D., 2006. *Kriptografi: Keamanan Data Dan Komunikasi*. 1st ed. Graha Ilmu.

Beaulieu, R., 2018. *Nsacyber/Simon-Speck*. [online] GitHub. Available at: <https://github.com/nsacyber/simon-speck> [Accessed 29 June 2018].

2009. [ebook] Available at: <https://library.binus.ac.id/eColls/eThesisdoc/Bab2DOC/2009-2-00611-IF%20BAB%202.doc> [Accessed 30 June 2018].

Denso, W., 2018. QR Code® Essentials. [ebook] Available at: <http://www.nacs.org/LinkClick.aspx?fileticket=D1FpVAvvJuo&tabid=1426&mid=4802> [Accessed 28 July 2018].

Fahmi, 2007. *Studi Dan Implementasi Watermarking Citra Digital Dengan Menggunakan Fungsi Hash*. [ebook] Available at: <http://file:///D:/NEW%20SKRIPPSI/skripsi%20ref%20qun/Makalah_TA%20Fahmi.pdf> [Accessed 30 June 2018].

Firmanesa, I., 2016. *Uji SAC Terhadap Algoritma Speck*. [ebook] Available at: <http://seminar.uny.ac.id/semnasmatematika/sites/seminar.uny.ac.id.semnasmatematika/files/T-15.pdf> [Accessed 30 June 2018].

Hastaka, A., 2017. *Pengamanan Soal Ujian Sekolah Dengan Algoritma Kriptografi Advance Encryption Standard (AES) Dan Metode Steganografi End Of File pada Sma Negeri 1 Weleri - Udinus Repository*. [online] Eprints.dinus.ac.id. Available at: <http://eprints.dinus.ac.id/22087/> [Accessed 30 June 2018].

Indriasari, T. and Rahayu, F., 2012. *Analisis Dan Perancangan Layanan Perpustakaan UAJY. 6-7.*. [ebook] Available at: <http://file:///C:/Users/Asus/Downloads/Documents/TF76301.pdf> [Accessed 30 June 2018].

Lanelo, B. and K Taniyo, A., 2011. *Analisis Penggunaan Steganografi Dengan Algoritma DES dan Fungsi Hash Untuk* Mengatasi Modifikasi *Citra*. [online] Scribd. Available at: <https://www.scribd.com/document/215299109/Analisis-Penggunaan-Steganografi-Dengan-Algoritma-DES-dan-Fungsi-Hash-Untuk-Mengatasi-Modifikasi-Citra> [Accessed 30 June 2018].

MItra, P. and Rakesh, N., 2016. *A Desktop Application Of QR Code For Data Security And Authentication. IEEE.*. [ebook] Available at: <http://file:///D:/NEW%20SKRIPPSI/skripsi%20ref%20qun/A%20Desktop%20Application%20of%20QR%20Code%20for%20Data%20Security%20and%20Authentication.pdf> [Accessed 30 June 2018].

Mohanty, B. and Narayan, M., 2018. A Novel SPECK Algorithm For Faster Image Compression - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: <http://ieeexplore.ieee.org/document/6918878/> [Accessed 28 July 2018].

Munir, R., 2006. *Kriptografi*.

Ngafifi, M., 2014. Kemajuan Teknologi Dan Pola Hidup Manusia Dalam Perspektif Sosial Budaya *Advances In Technology And Patterns Of Human Life In Socio-Cultural Perspective*. [online] Academia.edu. Available at: <http://www.academia.edu/34932243/Kemajuan_Teknologi_Dan_Pola_Hidup_Manusia_Dalam_PEerspektif_Sosial_Budaya_Advances_In_Technology_And_Patterns_Of_Human_Life_In_Socio-Cultural_Perspective> [Accessed 30 June 2018].

Qrcode.com. 2010. *About 2D Code| Qrcode.Com | DENSO WAVE*. [online] Available at: <http://www.qrcode.com/en/index.html> [Accessed 30 June 2018].

Qrcode.com. 2013. [online] Available at: <http://www.qrcode.com/> [Accessed 30 June 2018].

Sholeh, M. and Muharom, L., 2016. *Smart Presensi Menggunakan QR-Code Dengan Enkripsi Vigenere Cipher*. [online] Available at: <http://file:///D:/NEW%20SKRIPPSI/skripsi%20ref%20qun/Smart%20Presensi%20Menggunakan%20QR%20Code%20dengan%20Enkripsi%20Vigenere.pdf> [Accessed 30 June 2018].

Soewito, B., 2013. *Konsep Enkripsi Dan Dekripsi*. [ebook] Available at: <http://mti.binus.ac.id/files/2013/08/Konsep-Enkripsi-dekripsi.pdf> [Accessed 30 June 2018].

Syamsu, M., 2013. *Aspek Hukum Rahasia Bank Di Indonesia*. [ebook] Available at: <https://ejournal.unsrat.ac.id/index.php/lexprivatum/article/view/1012> [Accessed 30 June 2018].

Widiyanto, A., 2007. *Meningkatkan Keamanan Komputer Anda*. Semarang: Neomedia Press.